Web-ohjelmiston uuden version vaiheittainen käyttöönotto

Eveliina Pakarinen

Pro gradu -tutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 21. tammikuuta 2020

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Eveliina Pakarinen

Työn nimi — Arbetets titel — Title

Web-ohjelmiston uuden version vaiheittainen käyttöönotto

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	21. tammikuuta 2020	86

Tiivistelmä — Referat — Abstract

Tässä tutkielmassa ohjelmiston vaiheittaisella käyttöönotolla tarkoitetaan sitä, että ohjelmiston käyttäjille annetaan vaiheittain pääsy tuotantoon vietyyn ohjelmistoon. Otettaessa ohjelmistoa vaiheittain käyttöön annetaan tuotantoon vietyyn ohjelmistoon aluksi pääsy esimerkiksi vain tietylle osajoukolle ohjelmiston käyttäjistä. Tämän jälkeen kasvatetaan vaiheittain käyttäjien pääsyä ohjelmistoon. Vaiheittainen käyttöönotto päättyy, kun kaikille ohjelmiston käyttäjille on annettu pääsy ohjelmistoon.

Tässä tutkielmassa kehitettiin menetelmä web-ohjelmiston uuden version vaiheittaiselle käyttöönotolle ympäristöön, jossa vaatimuksena on palvelun käytön katkottomuus. Tutkielman tutkimuskysymyksenä on, miten web-ohjelmiston vaiheittainen käyttöönotto voidaan toteuttaa käytännössä. Tutkielma toteutettiin ohjelmistopalveluita tarjoavalle kohdeyritykselle, joka toimii yritykseltä-yritykselle-toimialalla. Tutkimusmenetelmänä tutkielmassa käytettiin design science -tutkimusmenetelmää.

Tutkielman aikana kehitettiin AphoDeploy-sovellus, jonka avulla vaiheittainen käyttöönotto voitiin automatisoida Kubernetes-klusterissa. Kubernetes-klusteriin asennettiin palveluverkkoratkaisu, jonka tarjoamien välityspalvelinten avulla Kubernetes-klusterissa tehtiin ajonaikaista liikenteenohjausta. AphoDeploy-sovelluksen avulla toteutettiin uuden ohjelmistoversion vaiheittainen käyttöönotto muuttamalla palveluverkkoratkaisun ja Kubernetes-klusterin konfiguraatioita automatisoidusti.

Vaiheittaisen käyttöönoton menetelmän toimintaa tutkittiin muun muassa automatisoiduilla testitapauksilla, joiden avulla selvitettiin, toteuttaako menetelmä vaiheittaiselle käyttöönotolle asetetut vaatimukset. Vaatimukset jaoteltiin teknisiin vaatimuksiin, vaiheittaisen käyttöönoton vaatimuksiin ja vaiheittaisen käyttöönoton päättymisen vaatimuksiin. Vaiheittaisen käyttöönoton menetelmän arvioinnin perusteella voitiin todeta, että kahdeksasta vaiheittaiselle käyttöönotolle asetetusta vaatimuksesta seitsemän vaatimusta täyttyi kokonaan ja yksi vaatimus täyttyi osittain.

Tutkielmassa löydettiin tapa toteuttaa web-ohjelmiston vaiheittainen käyttöönotto käytännössä AphoDeploy-sovelluksen avulla. Näin ollen tutkielmassa löydettiin vastaus tutkielman tutkimuskysymykseen. Tutkielman aikana kehitetyn menetelmän avulla luotiin lisäksi pohjaa työkaluinfrastruktuurille, joka voisi tulevaisuudessa tukea kohdeyrityksessä esimerkiksi tuotannossa tehtävää kokeilua. Vaiheittaisen käyttöönoton menetelmän arvioinnin perusteella AphoDeploy-sovelluksesta löydettiin tulevaisuutta varten myös mahdollisia jatkokehityskohteita.

ACM Computing Classification System (CCS):
· Software and its engineering → Software organization and properties → Contextual software domains → Software infrastructure → **Middleware**
· General and reference → Cross-computing tools and techniques → **Experimentation**

Avainsanat — Nyckelord — Keywords
jatkuva toimittaminen, ajonaikainen liikenteenohjaus, jatkuva kokeilu

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Esipuhe

Pro gradu -tutkielman kirjoittaminen on ollut pitkä urakka, jonka aikana on ehtinyt tapahtua paljon. Matkalla olen oppinut lisää monesta asiasta, ja kantavana voimana on ollut myös tutkielmassani esiintyvä toteamus[1]: ”kyllä se siitä”.

Tämä tutkielma on toteutettu Cinialle. Haluaisin kiittää Ciniaa opiskelujeni ja tutkielmani teon mahdollistamisesta näiden vuosien aikana. Lisäksi haluaisin kiittää ohjaajiani sekä Helsingin yliopistolla että Cinialla kaikista neuvoista ja pohdinnoista. Kiitos myös työkavereilleni kannustamisesta eteenpäin gradu-urakan kanssa.

Lopuksi haluaisin kiittää läheisiäni kaikesta tuesta, avusta ja kannustuksesta, joita olen heiltä vuosien varrella saanut. Vaikka pro gradu -tutkielman valmistumisen myötä yksi tavoite on nyt saavutettu, tulevaisuudessa vastaan tulee varmasti uusia mielenkiintoisia asioita.

Helsingissä, 21.01.2020
Eveliina Pakarinen

[1] http://motivaatiovalas.com/ (Vierailtu 20.12.2019).

Sisältö

1 Johdanto

Ohjelmistokehityksessä yksi ohjelmistokehityksen vaiheista on ohjelmiston käyttöönotto tuotannossa. Uuden ohjelmistoversion käyttöönottamiseksi on olemassa käytäntöjä, joiden avulla uusi ohjelmistoversio voidaan viedä tuotantoon. *Jatkuva integraatio* (*continuous integration, CI*), *jatkuva toimittaminen* (*continuous delivery, CDE*) ja *jatkuva julkaisu* (*continuous deployment, CD*) ovat käytäntöjä, joita käytetään nopeuttamaan ohjelmistojen kehittämistä ja tuotantoon viemistä (Humble & Farley, 2010).

Jatkuvassa integraatiossa tavoitteena on integroida ohjelmistokoodi vähintään muutaman kerran päivässä versionhallinnan päähaaraan, jotta päähaaraan integroitavat muutokset ja mahdolliset konfliktit ovat pieniä (Humble & Farley, 2010). Jatkuvassa toimittamisessa tavoitteena on, että jokainen laatuvaatimukset täyttävä versio ohjelmistosta on automaattisesti vietävissä tuotantoon nappia painamalla (Humble & Farley, 2010). Jatkuva julkaisu eroaa jatkuvasta toimittamisesta siten, että jatkuvassa julkaisussa ohjelmiston tuotantoon vienti tapahtuu automaattisesti, kun ohjelmiston uusi versio on läpäissyt automaattiset testit (Humble & Farley, 2010).

Jatkuva julkaisu tarjoaa mahdollisuuden automatisoituun ohjelmiston tuotantoon vientiin (Humble & Farley, 2010). Jatkuvan julkaisun avulla ohjelmistosta voidaan automaattisesti viedä tuotantoon vaihtoehtoinen toteutus, josta voidaan kerätä välittömästi palautetta ohjelmiston käyttäjiltä (Schermann, Cito, Leitner, Zdun, & Gall, 2018). Automatisoitu ohjelmiston tuotantoon vienti ja välitön palautteen kerääminen mahdollistavat tuotannossa tehtävät kokeilut, joiden avulla voidaan päättää, miten ohjelmiston kehitystä jatketaan (Schermann et al., 2018). *Jatkuvaksi kokeiluksi* (*continuous experimentation*) kutsutaan menetelmää, jossa ohjelmistokehitystä ohjaavat päätökset perustuvat dataan, jota kerätään tuotantoon viedystä ohjelmiston kokeellisesta versiosta (Schermann et al., 2018). Ohjelmiston kokeelliseen versioon voidaan muun muassa antaa pääsy vain osajoukolle kaikista ohjelmiston käyttäjistä.

Jatkuvaa julkaisua tutkittaessa yhdeksi konkreettiseksi tulevaisuuden tutkimuskohteeksi on mainittu tekninen infrastruktuuri, joka tukee jatkuvaa julkaisua (Rodríguez et al., 2017). Tekniikoita ohjelmiston uuden version viemiseksi tuotantoon on kuvattu ja esitelty tieteellisissä tutkimuksissa, mutta tekniikoiden käyttöä käytännössä ei ole käsitelty yksityiskohtaisella tasolla (Rodríguez et al., 2017).

Tässä tutkielmassa ohjelmiston vaiheittaisella käyttöönotolla tarkoitetaan sitä, että ohjelmiston käyttäjille annetaan vaiheittain pääsy tuotantoon vietyyn ohjelmistoon. Vaiheittaisesti käyttöönotettava ohjelmisto viedään aluksi tuotantoon, jonka jälkeen ohjelmistoon annetaan pääsy esimerkiksi vain tietylle osajoukolle ohjelmiston käyttäjistä. Tämän jälkeen kasvatetaan vaiheittain käyttäjien pääsyä ohjelmistoon. Vaiheittainen käyttöönotto päätty, kun kaikille ohjelmiston käyttäjille on annettu pääsy ohjelmistoon.

Tässä tutkielmassa selvitettiin, miten web-ohjelmiston uusi versio voidaan ottaa vaiheittain käyttöön ympäristössä, jossa vaatimuksena on palvelun käytön katkottomuus. Tähän tavoitteeseen liittyy tutkielman tutkimuskysymys:

> Miten web-ohjelmiston vaiheittainen käyttöönotto voidaan toteuttaa käytännössä?

Tutkielmassa etsitään ratkaisuja käytännön tekniikoiksi, joita voidaan käyttää web-ohjelmiston uuden version vaiheittaisessa käyttöönotossa.

Tutkielmassa tutkimusmenetelmänä käytetään *design science* -tutkimusmenetelmää (Hevner et al., 2004). Design science -menetelmässä luodaan ja arvioidaan tietoteknisiä artefakteja, jotka on tarkoitettu organisaatiollisten ongelmien ratkaisemiseen (Hevner et al., 2004). Design science -tutkimusmenetelmässä tutkimuksen osana suunnitellaan ja luodaan artefakti, jonka avulla havaittu tutkimusongelma voidaan ratkaista. Tässä tutkielmassa luotava artefakti on sovellus, jonka avulla web-ohjelmiston uusi versio voidaan ottaa vaiheittain käyttöön.

Tutkielman rakenne on seuraavanlainen: Luvussa 2 esitellään jatkuva ohjelmistotuotanto ja siihen liittyviä käytäntöjä. Luvussa 3 esitellään jatkuva kokeilu ja käytäntöjä tuotannossa suoritettavaan kokeiluun. Luvussa 4 käsitellään jatkuvan kokeilun käytäntöjen toteutustekniikoita ja työkaluja. Luvussa 5 esitellään tutkielman tutkimusmenetelmä ja tutkielmassa toteutettu vaiheittaisen käyttöönoton menetelmä. Luvussa 6 käydään läpi vaiheittaisen käyttöönoton menetelmän arviointi. Luvussa 7 esitellään arvioinnin tulokset ja tutkielman rajoitteet. Luvussa 8 esitellään tutkielman yhteenveto ja johtopäätökset.

2 Ohjelmistotuotannon jatkuvat käytännöt

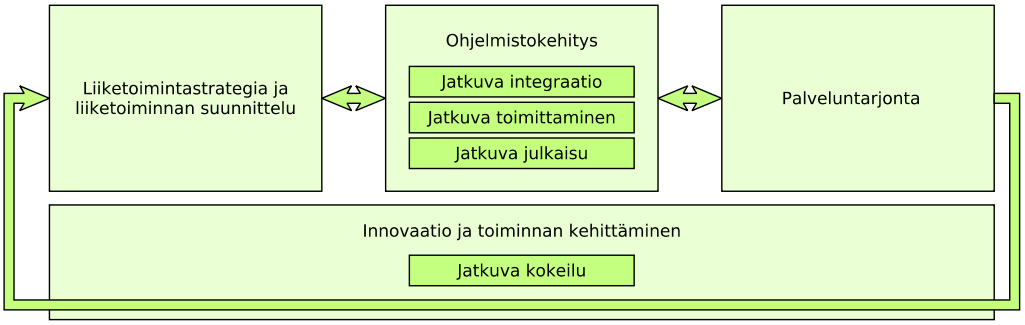
Seuraavissa aliluvuissa kuvataan jatkuvaa ohjelmistotuotantoa ja keinoja automatisoida ohjelmiston integraatio ja tuotantoon vienti. Ohjelmiston tuotantoon viennillä tarkoitetaan tässä tutkielmassa ohjelmistokehitysprosessin vaihetta, jossa kehityksessä ollut ohjelmisto siirretään tuotantoympäristöön, jonka kautta ohjelmisto on käytettävissä ohjelmiston käyttäjille.

2.1 Jatkuva ohjelmistotuotanto

Yksi tapa kuvata *jatkuvaa ohjelmistotuotantoa* (*continuous software engineering*) on jakaa ohjelmistotuotanto kolmeen osa-alueeseen, jotka ovat tiiviisti yhteydessä toisiinsa ja joiden välillä voidaan siirtyä osa-alueesta toiseen ilman katkoja ohjelmistotuotannon etenemisessä (Fitzgerald & Stol, 2017). Tässä kuvaustavassa ensimmäinen ohjelmistotuotannon osa-alueista on *liiketoimintastrategia ja liiketoiminnan suunnittelu* (*business strategy and planning*). Toisena osa-alueena on *ohjelmistokehitys* (*development*) ja kolmantena osa-alueena *palveluntarjonta* (*operations*). Lisäksi perustana jatkuvalle ohjelmistotuotannolle toimii *innovaatio ja toiminnan kehittäminen* (*innovation and improvement*).

Edellä mainittujen jatkuvan ohjelmistotuotannon osa-alueiden aikana voidaan toimia jatkuvien käytäntöjen mukaisesti (Fitzgerald & Stol, 2017). Yläkäsitteenä jatkuvan ohjelmistotuotannon käytännöille voidaan käyttää termiä *jatkuva ** eli *jatkuva tähti* (*continuous *, continuous star*) (Fitzgerald & Stol, 2017). Jatkuva * -termin alle voidaan sijoittaa nykyisin tunnistetut jatkuvat käytännöt, joita ovat esimerkiksi jatkuva integraatio, jatkuva toimittaminen ja jatkuva julkaisu. Lisäksi termiä voidaan käyttää yläkäsitteenä myös tulevaisuudessa tunnistettaville jatkuville käytännöille.

Kuvassa 2.1 esitellään jatkuvan ohjelmistotuotannon osa-alueiden suhteet toisiinsa ja siirtymät osa-alueiden välillä. Kuvaan on lisäksi sijoitettu tässä tutkielmassa käsiteltävät jatkuva * -käytännöt osaksi jatkuvan ohjelmistotuotannon osa-alueita. Tutkielmassa käsiteltävistä käytännöistä jatkuva integraatio, jatkuva toimittaminen ja jatkuva julkaisu sijoittuvat ohjelmistokehitysosa-alueen alle. Jatkuva kokeilu sijoittuu osa-alueen ”innovaatio ja toiminnan kehittäminen” alle.



Kuva 2.1: Tutkielmassa käsiteltävät jatkuva * -käytännöt osana jatkuvan ohjelmistotuotannon osa-alueita.

2.2 Jatkuva integraatio

Jatkuva integraatio (*continuous integration, CI*) on ohjelmistokehityskäytäntö, jossa ohjelmistokehityksen aikana tehdyt muutokset integroidaan usein versionhallinnan päähaaraan (Fowler, 2006). Tavoitteena on, että päähaaraan integroitavat muutokset ovat mahdollisimman pieniä, jotta myös konfliktit muutosten välillä ovat pieniä ja näin ollen myös helpommin löydettävissä ja korjattavissa. Jatkuvan integraation aikana ohjelmistokoodi käännetään automatisoidusti ja käännetty ohjelmisto testataan automatisoiduilla testeillä. Jos integroinnin aikana ohjelmiston kääntämisessä tai testauksessa ilmenee ongelmia, tulee ongelmat korjata mahdollisimman nopeasti, jotta versionhallinnan päähaarassa on aina saatavilla toimiva versio ohjelmistosta.

Jatkuvan integraation käytöstä voi olla hyötyä ohjelmistokehityksessä, mutta jatkuvan integraation käyttöä saattavat estää useat syyt. Yhtenä hyötynä jatkuvan integraation käytössä ohjelmistokehityksessä on, että jatkuva integraatio mahdollistaa ohjelmiston nopean julkaisun (Hilton, Tunnell, Huang, Marinov, & Dig, 2016). Lisäksi jatkuvaa integraatiota käytettäessä ohjelmistokehittäjät eivät ole yhtä huolissaan ohjelmiston käännöksessä esiintyvistä ongelmista kuin kehitettäessä ohjelmistoa ilman jatkuvaa integraatiota (Hilton et al., 2016).

Jatkuvan integraation käyttö vaatii kuitenkin ohjelmistokehittäjiltä taitoja käyttää jatkuvaa integraatiota osana ohjelmistokehitystä. Tällöin esteenä jatkuvan integraation käytölle voi olla, ettei ohjelmistokehittäjillä ole riittäviä taitoja jatkuvan integraation käyttämiselle ohjelmistokehityksessä (Hilton et al., 2016). Toinen este jatkuvan integraation käytölle voi myös olla automatisoitujen testien puute (Hilton et al., 2016).

2.3 Tuotantoon vienti jatkuvassa ohjelmistotuotannossa

Tuotantoon vientiä jatkuvassa ohjelmistotuotannossa voidaan kuvailla kahdella eri käsitteellä, joita ovat *jatkuva toimittaminen* (*continuous delivery, CDE*) ja *jatkuva julkaisu* (*continuous deployment, CD*). Näitä kahta käsitettä käytetään usein tieteellisessä kirjallisuudessa synonyymeinä (Rodríguez et al., 2017). Niiden määrittelystä voidaan kuitenkin löytää myös eroavaisuuksia. Seuraavissa alakohdissa esitellään jatkuvan toimittamisen ja jatkuvan julkaisun piirteitä. Lisäksi alakohdissa käsitellään jatkuvan toimittamisen -käsitteen ja jatkuvan julkaisun -käsitteen väliltä löytyviä eroja.

2.3.1 Jatkuva toimittaminen ja jatkuva julkaisu

Jatkuva toimittaminen on yksi jatkuva * -käytännöistä. Jatkuvassa toimittamisessa tavoitteena on, että ohjelmiston jokainen versio on vietävissä tuotantoon nappia painamalla (Humble & Farley, 2010). Edellytyksenä päätökselle viedä ohjelmiston uusi versio tuotantoon on, että ohjelmiston versio on läpäissyt automaattiset testit ja muut laadunvarmistustarkastukset (Humble, 2010). Jatkuvassa integraatiossa toteutetut asiat, kuten automatisoitu ohjelmiston kääntäminen ja automatisoidut testit, toimivat perustana jatkuvalle toimittamiselle (Humble & Farley, 2010).

Jatkuvalle julkaisulle ei ole tieteellisessä kirjallisuudessa muodostettu vakiintunutta määritelmää (Rodríguez et al., 2017). Jatkuvaa julkaisua kuvaillaan tieteellisessä kirjallisuudessa kuitenkin monella tavalla ja monesta eri näkökulmasta. Näistä kuvauksista on mahdollista löytää yhteisiä piirteitä, jotka määrittelevät teemat, joilla jatkuvan julkaisun eri osa-alueita kuvaillaan (Rodríguez et al., 2017). Jatkuvasta julkaisusta tehdyssä tutkimuksessa painottuu käytännön kokemusten esittely, sillä jatkuvaa julkaisua on kehitetty teollisuuslähtöisesti (Rodríguez et al., 2017).

Jatkuvassa julkaisussa ohjelmiston tuotantoon vienti tapahtuu automaattisesti ohjelmiston läpäistyä automaattiset testit (Humble, 2010). Tämä eroaa jatkuvasta toimittamisesta siten, että jatkuvassa toimittamisessa tuotantoon vienti tapahtuu manuaalisesti. Yhteinen piirre jatkuvalla toimittamisella ja jatkuvalla julkaisulla on, että molemmissa niistä ohjelmiston tulee olla täyttänyt laatuvaatimukset ennen kuin ohjelmisto viedään tuotantoon. Jatkuvan julkaisun voidaan siis ajatella olevan

automatisaatiolla laajennettu versio jatkuvasta toimittamisesta.

Jatkuvassa toimittamisessa ja jatkuvassa julkaisussa tuotantoon viennin aikatauluissa on myös eroja. Jatkuvassa julkaisussa tuotantoon viennin aikataulu määräytyy ohjelmistokehityksen etenemisen mukaan, sillä tuotantoon vienti tapahtuu automaattisesti ohjelmiston muuttuessa (Humble, 2010). Tämä eroaa jatkuvasta toimittamisesta siten, että jatkuvassa toimittamisessa tuotantoon viennin aikataulu voidaan määritellä liiketoiminnan tarpeiden mukaan (Humble, 2010).

Jatkuvan julkaisun mahdollistamiseksi ohjelmiston arkkitehtuurin on täytettävä tiettyjä piirteitä. Modulaarisuus ja löyhät kytkennät ohjelmiston osien välillä mahdollistavat, että ohjelmiston arkkitehtuuri on joustava ja sitä pystytään muuttamaan muuttuvien vaatimusten mukaan (Rodríguez et al., 2017). Siirryttäessä jatkuvasta integraatiosta jatkuvaan julkaisuun ohjelmiston arkkitehtuurin tulisi tukea ohjelmiston komponenttien itsenäistä tuotantoon vientiä ja tarjota mahdollisuus palauttaa komponentit aiempaan ohjelmistoversioon (Olsson, Bosch, & Alahyari, 2013).

Myös organisaatiokulttuuriin liittyy piirteitä, jotka vaikuttavat jatkuvan julkaisun käyttöön. Organisaatiokulttuuriin liittyviä pääpiirteitä ovat läpinäkyvyys organisaation sisällä, innovatiivinen ja kokeellisuuden mahdollistava organisaatiokulttuuri sekä yhteistyö organisaation eri osa-alueiden välillä (Rodríguez et al., 2017). Esimerkiksi yhteistyö ja vastuun jakaminen eri sidosryhmien, kuten ohjelmistokehittäjien, ohjelmistotestaajien, laadunvarmistustiimin ja operoinnin, välillä on edellytyksenä jatkuvalle julkaisulle (Shahin, Babar, & Zhu, 2016).

Organisaatiokulttuuriin liittyvän yhteistyön lisäksi myös palautteen kerääminen asiakkaalta on yksi jatkuvan julkaisun yleisesti mainituista piirteistä (Rodríguez et al., 2017). Asiakaspalautetta voidaan kerätä esimerkiksi tietyiltä henkilöiltä liittyen johonkin yksittäiseen asiaan tai ongelmaan ohjelmistossa. Jatkuvaa julkaisua käsittelevässä kirjallisuudessa ei kuitenkaan kerrota tarkasti, minkälaisia tapoja asiakaspalautteen käsittelemiseksi on olemassa tai miten asiakaspalaute huomioidaan ohjelmistokehityksen aikana (Rodríguez et al., 2017).

2.3.2 Tuotantoon viennin käytännöt

Tuotantoon viennin prosessi voidaan automatisoida käytännössä *tuotantoon viennin putken* (*deployment pipeline*) avulla (Humble & Farley, 2010). Tuotantoon viennin

putken avulla ohjelmistokoodiin tehdyn muutoksen laatu varmistetaan, jotta tiedetään, onko uusi versio ohjelmistosta riittävän hyvä vietäväksi tuotantoon. Tuotantoon viennin putki koostuu useasta vaiheesta, joiden aikana ohjelmiston laatu varmistetaan (Humble & Farley, 2010). Jos ohjelmiston uusi versio läpäisee kaikki vaiheet onnistuneesti, ohjelmiston uuden version voidaan todeta olevan riittävän hyvä vietäväksi tuotantoon.

Yksi tapa määritellä tuotantoon viennin putki on määritellä seitsemän osaa, joista putki voi muodostua, mutta joista kaikkia putken ei ole välttämätöntä sisältää (Shahin, Babar, & Zhu, 2017). Tässä määrittelytavassa tuotannon viennin putken osia ovat (Shahin et al., 2017):

1. versionhallintajärjestelmä,
2. ohjelmistokoodin hallinta- ja analysointityökalu,
3. ohjelmiston käännöstyökalu,
4. jatkuvan integraation palvelin,
5. ohjelmiston testaustyökalu,
6. ohjelmiston konfiguroinnin ja käyttöönoton valmistelun työkalut,
7. tuotantoon viennin palvelin.

Tuotantoon viennin putken osien toteuttamista varten löytyy myös useita työkaluja, joita voidaan hyödyntää käytännössä putken toteuttamisessa (Shahin et al., 2017).

Ohjelmistokehityksessä voidaan tehdä valinta, käytetäänkö ohjelmistokehityksen aikana jatkuvaa toimittamista vai jatkuvaa julkaisua. Monet organisaatiot tyytyvät jatkuvaan toimittamiseen jatkuvan julkaisun sijaan (Leppänen et al., 2015). Päätös olla siirtymättä jatkuvaan julkaisuun saattaa olla tietoinen tai johtua esteistä, jotka estävät automatisoidun ohjelmiston viennin tuotantoon (Leppänen et al., 2015). Asiakas saattaa olla tyytyväinen tuotantoon vienteihin, jotka tehdään usean kuukauden välein, tai asiakkaan toimialue voi rajoittaa automatisoitua tuotantoon vientiä (Leppänen et al., 2015).

Jatkuvan julkaisun käytöstä käytännössä on raportoitu tieteellisessä kirjallisuudessa useita hyötyjä ja haasteita (Rodríguez et al., 2017). Tieteellisessä kirjallisuudessa jatkuvan julkaisun hyödyistä on raportoitu usein ainoastaan ohjelmistoteollisuudessa tehdyissä raporteissa, tai väitettyjen hyötyjen taustoille ei ole annettu rationaalisia tai

tarkkoja selityksiä (Rodríguez et al., 2017). Tästä johtuen jatkuvan julkaisun käytön todellisista hyödyistä ei voi olla täyttä varmuutta (Rodríguez et al., 2017).

Jatkuvan julkaisun yksi usein mainittu hyödyllinen piirre on, että jatkuvaa julkaisua käyttävän organisaation on mahdollista julkaista uusia toiminnallisuuksia suoraan ohjelmiston käyttäjille nopeammin ja useammin kuin käytettäessä perinteistä ohjelmistokehitysprosessia (Rodríguez et al., 2017). Jatkuvan julkaisun käyttö on yleistä web-ohjelmistoja kehitettäessä, mutta sitä voidaan hyödyntää myös sulautettujen järjestelmien ja työpöytäohjelmistojen kehittämisessä (Rodríguez et al., 2017). Haasteita jatkuvaan julkaisuun siirtymisessä voivat aiheuttaa esimerkiksi tarvittavat muutokset organisaatiokulttuurissa tai liiketoimintastrategiassa (Rodríguez et al., 2017).

3 Tuotannossa suoritettava kokeilu

Jatkuvan toimittamisen tai jatkuvan julkaisun avulla voidaan ohjelmistosta viedä tuotantoon uusi versio automatisoidusti joko nappia painamalla tai automaattisesti, kun uusi ohjelmistoversio on läpäissyt laatuvaatimukset varmistavat testit. Tuotantoon viennin automatisaatio mahdollistaa sen, että tuotantoon voidaan viedä ohjelmistosta myös uusi kokeellinen versio tai vaihtoehtoinen toteutus, josta voidaan kerätä välittömästi palautetta ohjelmiston käyttäjiltä (Schermann et al., 2018).

Ohjelmiston käyttäjiltä kerättävä palaute ja kokeellisten versioiden vieminen tuotantoon mahdollistavat tuotannnossa suoritettavat kokeilut (Schermann et al., 2018). Yksi tuotannossa suoritettaviin kokeiluihin liittyvistä jatkuva * -käytännöistä on *jatkuva kokeilu* (*continuous experimentation, CE*). Seuraavissa aliluvuissa käsitellään jatkuvan kokeilun määritelmää ja käytäntöjä.

3.1 Jatkuva kokeilu

Jatkuvan julkaisun tapaan myös jatkuvaa kokeilua kuvataan tieteellisessä kirjallisuudessa usealla eri käsitteellä (Auer & Felderer, 2018). Yksi jatkuvan kokeilun määritelmistä on, että jatkuva kokeilu on ohjelmistokehitysmenetelmä, jossa ohjelmistokehitystä ohjaavat päätökset perustuvat dataan, jota kerätään tuotantoon viedystä ohjelmiston kokeellisesta versiosta (Schermann et al., 2018).

Toisena määritelmänä jatkuvalle kokeilulle on sijoittaa jatkuva kokeilu osaksi mallia, jossa kuvataan jatkuva * -käytäntöjen suhdetta toisiinsa. Jatkuva integraatio, jatkuva toimittaminen ja jatkuva julkaisu voidaan nähdä vaiheina, jotka ovat osa ohjelmistokehityksen muuttumista kohti innovaatioihin ja kokeiluun perustuvaa ohjelmistokehitystapaa (Olsson et al., 2013). Tässä mallissa viimeisenä vaiheena on siirtyminen jatkuvan julkaisun käytöstä hyödyntämään tuotannossa tehtäviä kokeiluja ja kokeiluista kerättävää palautetta ohjelmistokehityksen ohjaamisessa (Olsson et al., 2013).

Jatkuvan kokeilun tavoitteena voi olla löytää teknisiä ongelmia ohjelmiston uudesta versiosta tai selvittää ohjelmistoon toteutetun uuden ominaisuuden liiketoiminnallinen arvo (Schermann et al., 2018). *Regressiopohjaisessa jatkuvassa kokeilussa* (*regression-driven experiments*) keskitytään teknisten ongelmien etsimiseen ja *liiketoimintapohjaisessa jatkuvassa kokeilussa* (*business-driven experiments*) keskitytään uusien ominaisuuksien liiketoiminnallisen arvon selvittämiseen (Schermann et al., 2018).

Regressiopohjainen jatkuva kokeilu: Regressiopohjaisessa jatkuvassa kokeilussa kokeilujen tavoitteena on pienentää uuden ohjelmistoversion asennukseen liittyvää teknistä riskiä ja varmistaa uuden version oikea toiminta (Schermann et al., 2018). Ongelmat, joita regressiopohjaisen jatkuvan kokeilun avulla voidaan etsiä, voivat liittyä esimerkiksi ohjelmiston toiminnallisuuteen tai suorituskykyyn. Regressiopohjaisessa jatkuvassa kokeilussa kokeilut ovat usein lyhytaikaisia kokeilujen pituuden vaihdellessa muutamasta minuutista muutamiin päiviin. Aluksi kokeiluun valitaan usein mukaan vain pieni joukko ohjelmiston käyttäjiä, mutta kokeilun edetessä lisää käyttäjiä otetaan mukaan kokeiluun. Regressiopohjaisen jatkuvan kokeilun käyttö ei kuitenkaan ole kovin yleistä käytännössä.

Liiketoimintapohjainen jatkuva kokeilu: Liiketoimintapohjaisessa jatkuvassa kokeilussa tavoitteena on selvittää uusien ominaisuuksien liiketoiminnallinen arvo (Schermann et al., 2018). Liiketoimintapohjaisessa jatkuvassa kokeilussa kerätään kokeilun aikana dataa, jota analysoidaan usein tarkemmin kuin dataa, jota kerätään regressiopohjaisen jatkuvan kokeilun aikana. Liiketoimintapohjaiset jatkuvat kokeilut kestävät myös pidempään kuin regressiopohjaiset jatkuvat kokeilut, sillä liiketoimintapohjaisten jatkuvien kokeilujen pituus on usein useita viikkoja. Pitkien

kokeilujen tavoitteena on kerätä riittävästi dataa, jonka perusteella on mahdollista tehdä tilastollisesti merkittäviä johtopäätöksiä.

Jatkuvan kokeilun käyttöönottoa ohjelmistoteollisuudessa vaikeuttavat monet haasteet. Yksi suurimmista haasteista jatkuvan kokeilun käyttöönotolle on organisaatiokulttuuri sekä yleisesti ohjelmistokehitysorganisaatioissa että erityisesti *yritykseltä-yritykselle-toimialalla* (*business-to-business, B2B*) asiakasorganisaatioissa (Lindgren & Münch, 2016). Organisaatiossa ongelmakohtina voivat olla esimerkiksi riittämätön ketteryys tai läpinäkyvyys tai ongelmat sidosryhmien yhteistyössä (Lindgren & Münch, 2016). Myös ajallisten ja taloudellisten resurssien puute sekä vaikeus mitata kokeilujen asiakkaille tuottamaa arvoa sopivien mittareiden avulla ovat haasteita jatkuvan kokeilun käytössä ohjelmistoteollisuudessa (Lindgren & Münch, 2016). Yritykseltä-yritykselle-toimialalla haasteena voi olla myös, että ohjelmiston käytöstä ei aina saada suoraan palautetta ohjelmiston loppukäyttäjiltä, mikä vaikeuttaa palautteen keräämistä (Rissanen & Münch, 2015).

Haasteista huolimatta ohjelmistoteollisuudessa esiintyy kiinnostusta jatkuvaa kokeilua kohtaan (Lindgren & Münch, 2016). Jatkuva kokeilu ei kuitenkaan ole ohjelmistoteollisuudessa laajassa käytössä eikä vakiintunut osa ohjelmistokehitysprosessia (Lindgren & Münch, 2016). Ohjelmistoteollisuuden toimintatavoissa tulisi tapahtua jatkuvaa kokeilua tukevia muutoksia, jotta jatkuva kokeilu voisi tulla laajempaan käyttöön ohjelmistoteollisuudessa. Muutoksia voitaisiin tehdä muun muassa yrityskulttuurissa, liiketoimintatavoissa, jatkuvaa kokeilua tukevan ohjelmistoarkkitehtuurin kehittämisessä ja menetelmissä kerätä ohjelmiston käytöstä tietoa (Olsson et al., 2013).

Jatkuvaa kokeilua tukeva ohjelmistoarkkitehtuuri ja ohjelmiston toiminnan seuraaminen valvonnan avulla ovat teknisiä edellytyksiä jatkuvalle kokeilulle (Schermann et al., 2018). Jatkuvassa kokeilussa ohjelmistokehitys pohjautuu ohjelmiston käytöstä kerättyyn palautteeseen. Tästä syystä ohjelmistosta on pystyttävä keräämään tietoa siitä, miten ohjelmistoa käytetään käytännössä (Olsson et al., 2013). Jotta jatkuva kokeilu olisi mahdollista, ohjelmiston tulee koostua itsenäisesti julkaistavista komponenteista, joiden tuotantoon vienti on automatisoitu (Schermann et al., 2018). Kokeiluja voidaan tehdä myös ohjelmiston suorituksen aikana, jos ohjelmiston arkkitehtuuri mahdollistaa ohjelmiston toiminnallisuuden muuttamisen suorituksen aikana

(Olsson et al., 2013).

Myös yrityksen organisaation ja yrityskulttuurin on tuettava jatkuvaa kokeilua (Schermann et al., 2018). Muun muassa yrityksen liiketoimintamallin tulee tukea lyhyessä ajassa tapahtuvaa ohjelmistokehitystä (Olsson et al., 2013). Tiedon jakaminen kaikille ohjelmistokehitysprosessiin osallistuville henkilöille on myös tärkeä yrityskulttuuriin liittyvä piirre, joka tukee jatkuvaa kokeilua (Schermann et al., 2018).

Jatkuvaa kokeilua käsittelevän tieteellisen kirjallisuuden määrä on kasvanut jatkuvasti vuodesta 2010 alkaen (Auer & Felderer, 2018). Lisäksi jatkuvaa kokeilua käsittelevässä tieteellisessä kirjallisuudessa tehdään paljon yhteistyötä akatemian ja teollisuuden välillä. Yhteistyö akatemian ja teollisuuden välillä näkyy esimerkiksi siinä, että tieteellisessä tutkimuksessa käytetään tutkimusmenetelmiä, joissa tutkimustulokset arvioidaan käytännössä simuloinnin sijaan. Tieteellisessä kirjallisuudessa ei kuitenkaan esitellä usein ohjeita tai työkaluja, joiden avulla jatkuva kokeilu on käytännössä toteutettu.

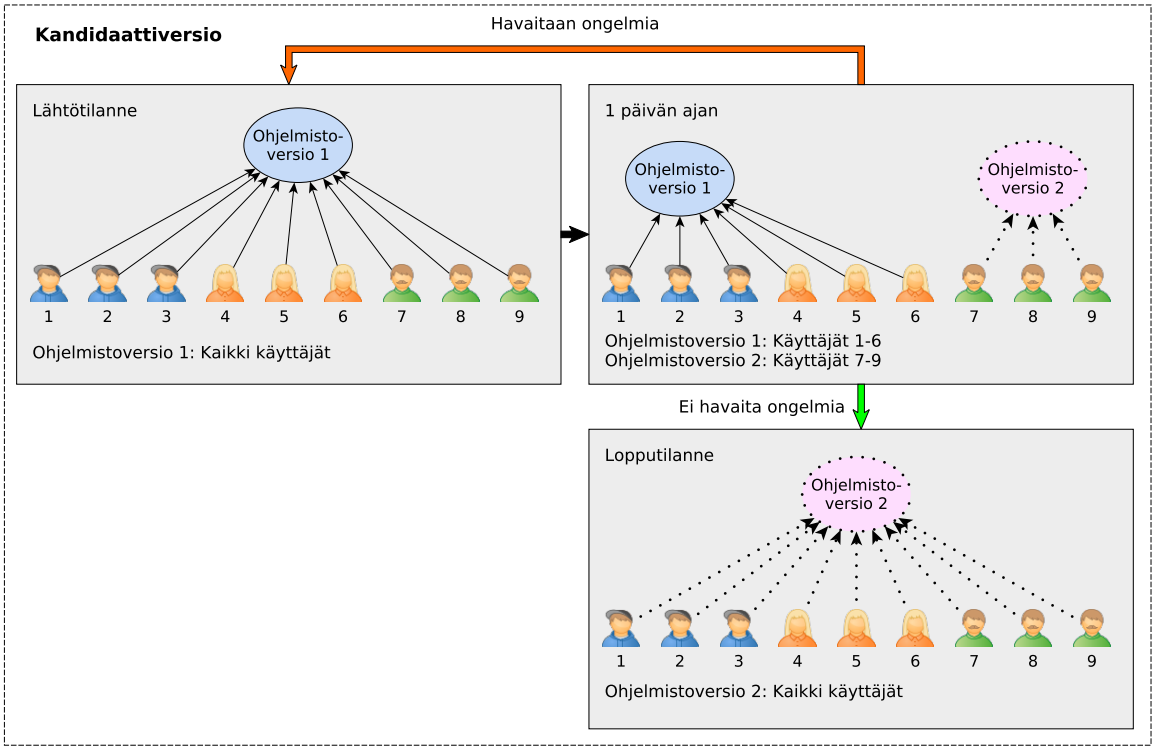
Jatkuvaa kokeilua tukevat työkalut ja työkaluinfrastruktuuri ovat kuitenkin oleellisia edellytyksiä jatkuvalle kokeilulle (Lindgren & Münch, 2016). Jatkuvaa kokeilua tukevan työkaluinfrastruktuurin luominen on yksi käytännön asia, jolla jatkuva kokeilu voidaan mahdollistaa. Työkaluinfrastruktuurin kehittäminen on siis yksi mahdollinen jatkuvaan julkaisuun liittyvistä tulevaisuuden tutkimuskohteista.

3.2 Jatkuvan kokeilun käytännöt

Jatkuvaa kokeilua voidaan tehdä käyttäen useaa käytäntöä, joita ovat muun muassa *kandidaattiversio* (*canary release*) ja *asteittainen julkistaminen* (*gradual rollout*) (Schermann et al., 2018).

Kandidaattiversio: Kandidaattiversio on käytäntö, jossa tuotantoympäristöön viedään ohjelmistosta päivitetty versio, joka otetaan käyttöön vain pienelle joukolle ohjelmiston käyttäjistä (Humble & Farley, 2010). Kandidaattiversion avulla voidaan uuden ohjelmistoversion toimintaa kokeilla tuotantoympäristössä ja varmistaa, että mahdolliset ongelmat koskevat vain pientä osaa ohjelmiston käyttäjistä (Humble & Farley, 2010). Kuvassa 3.1 nähdään esimerkki kandidaattiversio-käytännön toiminnasta. Lähtötilanteessa kaikki ohjelmiston käyttäjät ohjataan ohjelmistoversioon 1. Seuraavassa vaiheessa pieni osa käyttäjistä ohjataan käyttämään ohjelmistoversiota 2.

Tämän jälkeen, jos ohjelmistoversion 2 toiminnassa havaitaan ongelmia, voidaan palata takaisin lähtötilanteeseen. Vaihtoehtoisesti, jos ohjelmistoversio 2 toimii odotusten mukaan, voidaan kaikki ohjelmiston käyttäjät ohjata käyttämään ohjelmistoversiota 2.

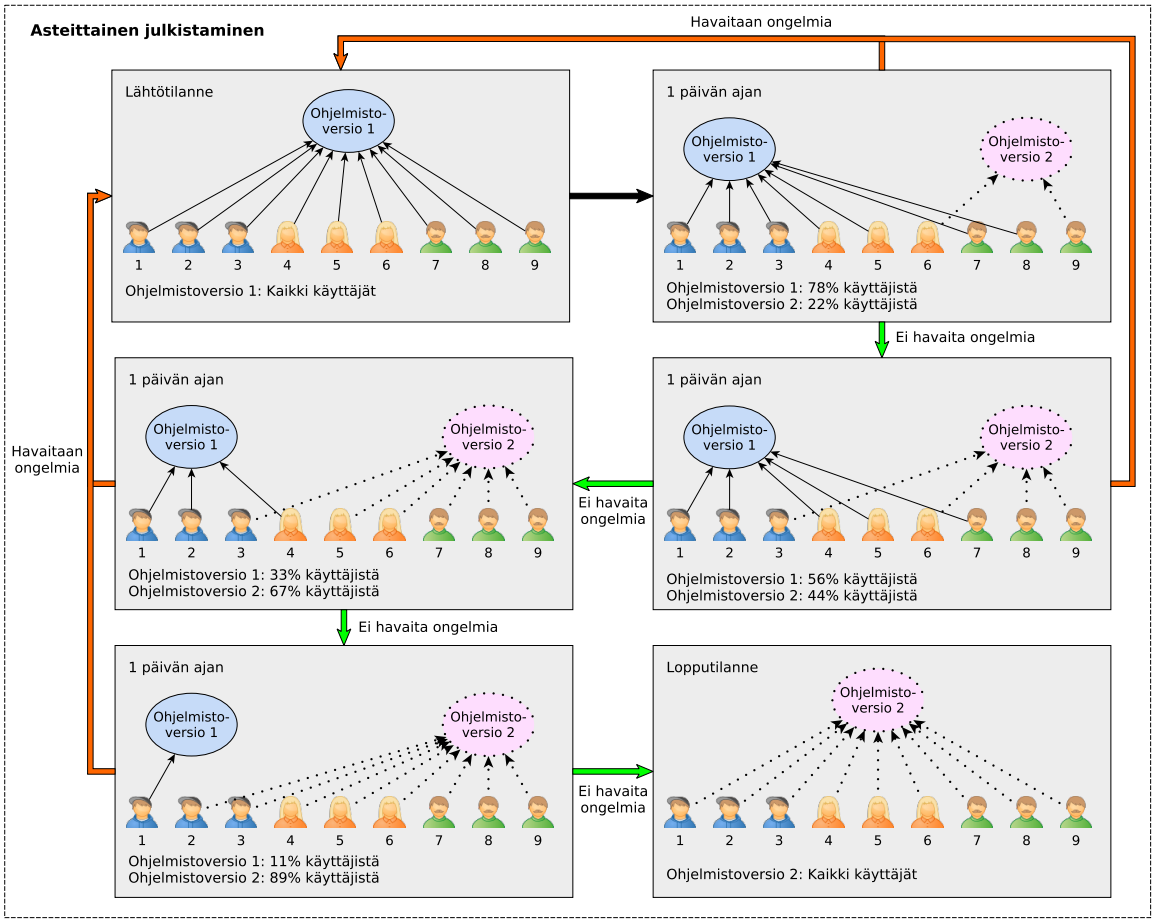


Kuva 3.1: Esimerkki kandidaattiversio-käytännön käytöstä.

Kandidaattiversiota käytetään usein regressiopohjaisen jatkuvan kokeilun toteuttamiseen (Schermann et al., 2018). Haasteena kandidaattiversion käytössä ovat ohjelmiston jaetut resurssit kuten tietokanta tai yhteydet ulkoisiin palveluihin (Humble & Farley, 2010). Jaettujen resurssien on toimittava kaikkien ohjelmiston versioiden kanssa, jos tuotantoympäristössä otetaan käyttöön useita versioita ohjelmistosta (Humble & Farley, 2010). Kandidaattiversion kautta jaettuihin tietoihin tehtyjen muutosten tulee myös olla taaksepäinyhteensopivia muun järjestelmän kanssa (Neely & Stolt, 2013).

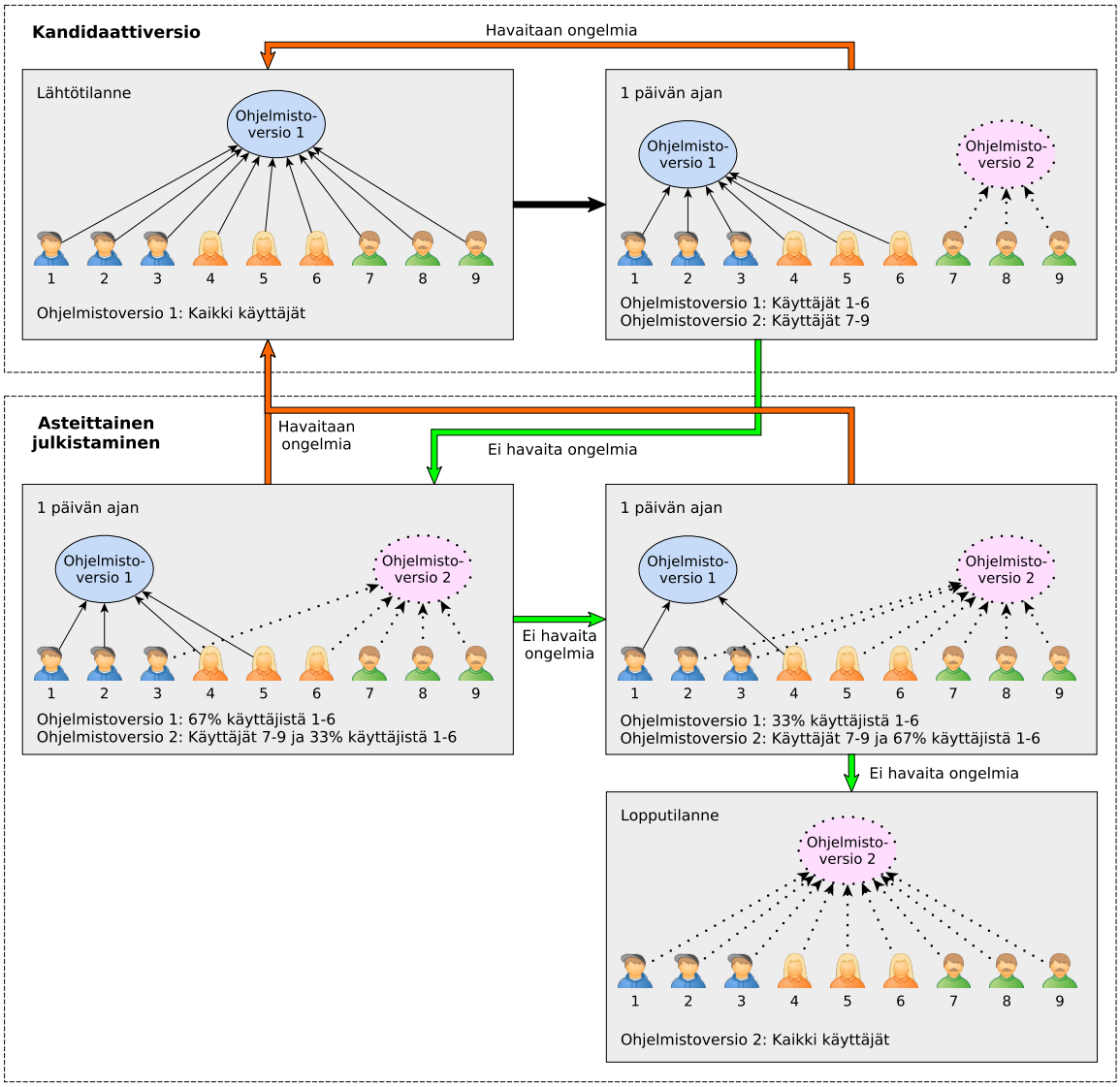
Asteittainen julkistaminen: Kandidaattiversio-käytäntöön voidaan yhdistää myös asteittainen julkistaminen, jossa ohjelmiston uuteen versioon ohjataan vaiheittain lisää käyttäjiä, kunnes kaikki käyttäjät tai ennalta määrätty osa käyttäjistä on ohjattu käyttämään uutta ohjelmistoversiota (Humble & Farley, 2010). Asteit-

taista julkistamista voidaan myös käyttää regressiopohjaisessa jatkuvassa kokeilussa (Schermann et al., 2018). Kuvassa 3.2 nähdään esimerkki asteittaisen julkistamisen käytöstä. Asteittaista julkistamista käytettäessä uuteen ohjelmistoversioon 2 ohjataan vaiheittain enemmän käyttäjiä, kunnes kaikki käyttäjät on ohjattu käyttämään uutta ohjelmistoversiota. Ongelmatilanteissa voidaan ohjata kaikki käyttäjät ohjelmistoversioon 1.



Kuva 3.2: Esimerkki asteittaisen julkistamisen käytöstä.

Kuvassa 3.3 nähdään yhdistelmä kandidaattiversion ja asteittaisen julkistamisen käytöstä. Yhdistelmässä voidaan ensin kandidaattiversio-käytännön avulla kokeilla uutta ohjelmistoversiota tietyllä osalla kaikista käyttäjistä. Tämän jälkeen voidaan vaiheittain ohjata lisää käyttäjiä käyttämään uutta ohjelmistoversiota asteittaisen julkistamisen avulla.



Kuva 3.3: Esimerkki kandidaattiversion ja asteittaisen julkistamisen yhdistelmän käytöstä.

4 Tuotannossa kokeilun tekniikat ja työkalut

Kaksi yleistä tekniikkaa, joilla jatkuvan kokeilun käytäntöjä voidaan toteuttaa, ovat *ominaisuusmuuttujat* (*feature toggles*) ja *ajonaikainen liikenteenohjaus* (*runtime traffic routing*) (Schermann et al., 2018). Tässä luvussa käsitellään tuotannossa kokeilun tekniikoita ja työkaluja. Lisäksi luvussa esitellään tekniikka tuotannossa suoritettavan kokeilun automatisoimiseksi.

4.1 Ominaisuusmuuttujat

Ominaisuusmuuttujat ovat muuttujia, joiden avulla voidaan lähdekoodissa ehdollisesti sallia tai estää pääsy tiettyyn ominaisuuteen (Rahman, Querel, Rigby, & Adams, 2016). Ominaisuusmuuttujat voivat mahdollistaa ohjelmiston toiminnallisuuden muuttamisen myös ajonaikaisesti, jos ominaisuusmuuttujien avulla sallitaan esimerkiksi tietylle käyttäjäryhmälle pääsy vaihtoehtoiseen suorituspolkuun ohjelmistossa (Hodgson, 2017).

Ominaisuusmuuttujat ovat yksinkertaisimmillaan ehdollisia if-else-lauseita (Hodgson, 2017). Esimerkissä 4.1 esitellään rivillä 3 ominaisuusmuuttuja `enhancedVersion`. Ominaisuusmuuttujalla määritellään `getAphorism`-metodissa, mikä aforismi metodista palautetaan. Jos `enhancedVersion`-muuttuja on käytössä eli sen arvo on `true`, palauttaa metodi aforismin `enhancedAphorism` rivillä 8. Muussa tapauksessa metodi palauttaa aforismin `basicAphorism` rivillä 11. Esimerkistä 4.1 huomataan, että ominaisuusmuuttujan `enhancedVersion` käyttö vaikuttaa ohjelmiston lähdekoodiin. Tällöin myös ominaisuusmuuttujan muuttamisesta tai poistamisesta seuraa muutoksia ohjelmiston lähdekoodiin.

```
1  public class AphorismService {
2      @Value("${enhanced.version}")
3      private boolean enhancedVersion;
4
5      public AphorismDto getAphorism(){
6          if (enhancedVersion) {
7              AphorismDto enhancedAphorism = new AphorismDto("Elä niin kuin kuolisit
                   huomenna. Opi niin kuin eläisit ikuisesti. - Mahatma Gandhi",
                   "Oppiminen ja eläminen");
8              return enhancedAphorism;
9          } else {
10             AphorismDto basicAphorism = new AphorismDto("Kyllä se siitä -
                   Motivaatiovalas", "Motivoiva toteamus");
11             return basicAphorism;
12         }
13     };
14 };
```

Esimerkki 4.1: Toiminnallisuuden muuttaminen ominaisuusmuuttujan avulla.

Ominaisuusmuuttujat voidaan jaotella kategorioihin ominaisuusmuuttujan käyttötarkoituksen mukaan (Hodgson, 2017). Ominaisuusmuuttujien avulla voidaan uuden keskeneräisen ominaisuuden sisältävä ohjelmistoversio viedä tuotantoon siten, että

pääsy keskeneräiseen ominaisuuteen estetään ominaisuusmuuttujan avulla (Rahman et al., 2016). Tällöin uuden ominaisuuden tullessa valmiiksi, voidaan ominaisuusmuuttuja poistaa ohjelmiston lähdekoodista (Rahman et al., 2016). Ominaisuusmuuttujien avulla voidaan myös rajoittaa valmiin ominaisuuden näkyvyys vain tietyille käyttäjille (Hodgson, 2017).

Toinen tapa käyttää ominaisuusmuuttujia on käyttää niitä uusien ominaisuuksien julkistamisen apuna (Rahman et al., 2016). Ominaisuusmuuttujien avulla voidaan uusi ominaisuus julkistaa asteittain tai peruuttaa julkistaminen ajonaikaisesti, jos ohjelmiston toiminnassa havaitaan ongelmia (Rahman et al., 2016). Ominaisuusmuuttujia voidaan siis käyttää tuotannossa tehtävän kokeilun toteutustekniikkana (Hodgson, 2017).

Ominaisuusmuuttujien käyttöön liittyy kuitenkin haasteita. Jos keskeneräisiä ominaisuuksia piilottavia ominaisuusmuuttujia jää ohjelmiston lähdekoodiin myös silloin, kun niiden piilottama ominaisuus on tullut valmiiksi, aiheutuu siitä teknistä velkaa ohjelmiston lähdekoodiin (Rahman et al., 2016). Lisäksi ominaisuusmuuttujat lisäävät ohjelmistokoodin monimutkaisuutta (Hodgson, 2017). Ominaisuusmuuttujien käyttö mahdollistaa sen, että ohjelmisto voi toimia usealla eri tavalla riippuen siitä, mikä ominaisuusmuuttuja on käytössä. Tällöin, mitä enemmän ominaisuusmuuttujia on käytössä, sitä useammalla eri tavalla ohjelmisto voi toimia. Ohjelmiston toiminnan monimutkaistuminen voi siis aiheuttaa paljon lisätyötä ohjelmiston toiminnan testaamisessa (Hodgson, 2017).

4.2 Ajonaikainen liikenteenohjaus

Ajonaikaisella liikenteenohjauksella (*runtime traffic routing*) tarkoitetaan verkkoliikenteen ohjaamista ohjelmiston suorituksen aikana tiettyyn ohjelmiston versioon esimerkiksi välityspalvelimen avulla (Schermann et al., 2018). Liikenteen ohjaamista voidaan tehdä esimerkiksi HTTP-pyyntöjen otsakkeiden[2] arvojen perusteella.

Suoritettaessa ohjelmistosta useita versioita yhtäaikaisesti, voidaan liikennettä ohjata ohjelmiston eri versioiden välillä. Ohjelmistosta voidaan suorittaa useita kontitettuja versioita yhtä aikaa muun muassa Kubernetes-klusterissa[3] (The Kubernetes

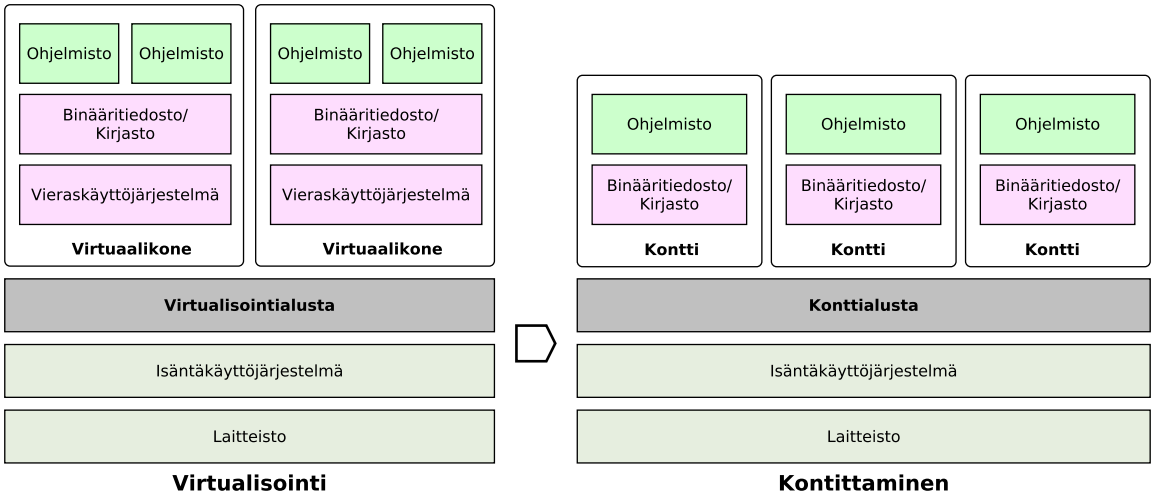
[2]https://tools.ietf.org/html/rfc7230 (Vierailtu 26.12.2019).
[3]https://kubernetes.io/ (Vierailtu 28.12.2019).

Authors, 2019d). Tässä luvussa esitellään kontteihin ja konttien hallinnointijärjestelmiin liittyviä peruskäsitteitä. Lisäksi luvussa esitellään palveluverkkoratkaisu Istio[4], jonka avulla ajonaikaista liikenteenohjausta voidaan tehdä käytännössä Kubernetes-klusterissa.

4.2.1 Kontit ja konttien hallinnointijärjestelmät

Kontteihin läheisesti liittyvä käsite on virtualisointi (The Kubernetes Authors, 2019i). Virtualisoinnin avulla voidaan yhdessä laitteistossa suorittaa yhtäaikaisesti useita *virtuaalikoneita* (*virtual machine*) yhden isäntäkäyttöjärjestelmän päällä. Virtuaalikoneita suoritetaan isäntäkäyttöjärjestelmässä *virtualisointialustan* (*hypervisor*) avulla. Virtuaalikoneet ovat eristettyjä ympäristöjä, jotka muodostuvat virtuaalikoneessa suoritettavista ohjelmistoista, binääritiedostoista ja kirjastoista sekä virtuaalikoneen vieraskäyttöjärjestelmästä.

Konttien (*container*) avulla voidaan paketoida ohjelmiston lähdekoodi ja ohjelmiston riippuvuudet yhdeksi kokonaisuudeksi (Docker Inc., 2019). Virtuaalikoneista poiketen kontit eivät sisällä vieraskäyttöjärjestelmää, vaan kontit jakavat yhden isäntäkäyttöjärjestelmän, jonka päälle asennetulla konttialustalla kontteja suoritetaan (The Kubernetes Authors, 2019i). Kuvassa 4.1 esitellään kontitetun ja virtualisoidun ympäristön arkkitehtuurit. Kuvasta nähdään myös virtuaalikoneiden ja konttien erot.



Kuva 4.1: Virtualisoinnin ja kontittamisen esittely (The Kubernetes Authors, 2019i).

[4]https://istio.io/ (Vierailtu 02.12.2019).

Konttien automaattista hallintaa varten on kehitetty hallinnointijärjestelmiä, joiden avulla voidaan muun muassa hallita konteista muodostettuja klustereita. Yksi konttien hallinnointijärjestelmistä on Googlen kehittämä Kubernetes (Verma et al., 2015). Avoimella lähdekoodilla toteutettu Kubernetes perustuu Googlen kehittämään suljetun lähdekoodin Borg-järjestelmään (Verma et al., 2015). Kuberneteksen kehityksessä on otettu huomioon Borg-järjestelmässä havaittuja hyviä ominaisuuksia ja korjattu Borg-järjestelmässä havaittuja ongelmia (Verma et al., 2015).

Käytettäessä Kubernetesta konttien hallinnointijärjestelmänä suoritetaan Kuberneteksen hallinnoimia kontitettuja ohjelmistoja klusterissa, joka muodostuu solmuista eli yksittäisistä koneista (The Kubernetes Authors, 2019c). Kubernetes-klusteri muodostuu vähintään yhdestä *hallintasolmusta* (*master node*) ja vähintään yhdestä *kuormasolmusta* (*worker node*). Hallintasolmulle asennettujen komponenttien avulla hallitaan Kubernetes-klusteriin kuuluvia kuormasolmuja ja kuormasolmuilla suoritettavia kontitettuja ohjelmistoja.

Kuberneteksen toimintaperiaate perustuu ajatukseen *kontrollista koreografian avulla* (*control through choreography*) (Burns, Grant, Oppenheimer, Brewer, & Wilkes, 2016). Kubernetes-klusterin tilan hallinnointi perustuu klusterin halutun tilan ja klusterin nykyisen tilan vertaamiseen. Jos klusterin haluttu tila ja nykyinen tila eroavat, tehdään Kubernetes-klusterissa toimenpiteitä klusterin nykyisen tilan muuttamiseksi halutun tilan kaltaiseksi. Tämä tarkoittaa, että Kubernetes-klusterin tila muotoutuu vaiheittain itsenäisten komponenttien tekemien toimenpiteiden yhdistetyn vaikutuksen seurauksena.

Kubernetes-klusterin tilaa kuvataan Kubernetes-objektien avulla (The Kubernetes Authors, 2019h). Kubernetes-objektien avulla voidaan muun muassa määrittää, mitä kontitettuja ohjelmistoja Kubernetes-klusterissa suoritetaan ja kuinka paljon resursseja kontitetuilla ohjelmistoilla on käytettävissä (The Kubernetes Authors, 2019h). Kubernetes-klusterin tilan hallinnointiin käytetään Kubernetes-kontrollereita, joiden avulla klusterin nykyistä tilaa muutetaan tarvittaessa kohti klusterin haluttua tilaa (The Kubernetes Authors, 2019a). Taulukossa 4.1 esitellään kuvaukset yleisille Kubernetes-objekteille, joita ovat muun muassa Pod, Service ja nimiavaruus, ja yleiselle Kubernetes-kontrollerille, jota kutsutaan nimellä Deployment.

Kubernetes-termi	**Kuvaus**
Pod	Podit ovat pienimpiä Kubernetes-klusterissa suoritettavia yksiköitä, joiden sisällä voidaan suorittaa yhtä tai useampaa ohjelmistokonttia ja joiden avulla määritellään konttien suoritukseen liittyviä konfiguraatioita (The Kubernetes Authors, 2019f). Suoritettaessa Podin sisällä useaa läheisesti toisiinsa liittyvää konttia toinen konteista voi muun muassa täydentää toisen kontin toiminnallisuutta. Lisätoiminnallisuutta tarjoavaa konttia kutsutaan *sivuvaunuksi* (*sidecar*). Koska yhdessä Podissa suoritetaan yhtä instanssia ohjelmistosta, voidaan ohjelmistoinstanssien määrää skaalata replikoimalla Podeja eli kasvattamalla Podien määrää. Podien replikointia voidaan tehdä Kubernetes-kontrollerien avulla.
Deployment	Deployment-kontrollerin avulla voidaan keskitetysti hallinnoida replikoituja Podeja (The Kubernetes Authors, 2019b). Deploymentin avulla voidaan muun muassa skaalata Deploymentin hallinnoimien Podien määrää joko suuremmaksi tai pienemmäksi tai palata käyttämään aiempaa Deploymentin versiota, jos nykyisen Deploymentin version tila on epävakaa.
Service	Service-objektien avulla voidaan replikoituja Podeja ryhmitellä loogisesti kokonaisuuksiksi, joiden kautta Podien tarjoamia palveluita voidaan käyttää (The Kubernetes Authors, 2019g). Servicen kautta Podien tarjoamaa palvelua voidaan käyttää yhtenäisellä tavalla myös silloin, kun Servicen määrittelemään kokonaisuuteen kuuluvien Podien määrä muuttuu.
Nimiavaruus (*Namespace*)	Nimiavaruuksien avulla voidaan yksi fyysinen Kubernetes-klusteri jakaa useaan virtuaaliseen klusteriin (The Kubernetes Authors, 2019e). Virtuaalisissa klustereissa on muun muassa mahdollista käyttää Kubernetes-objekteilla samoja nimiä. Lisäksi Kubernetes-klusterissa voidaan määritellä nimiavaruuskohtaisesti, kuinka paljon fyysisen klusterin resursseja yhdellä nimiavaruudella on käytettävissä.

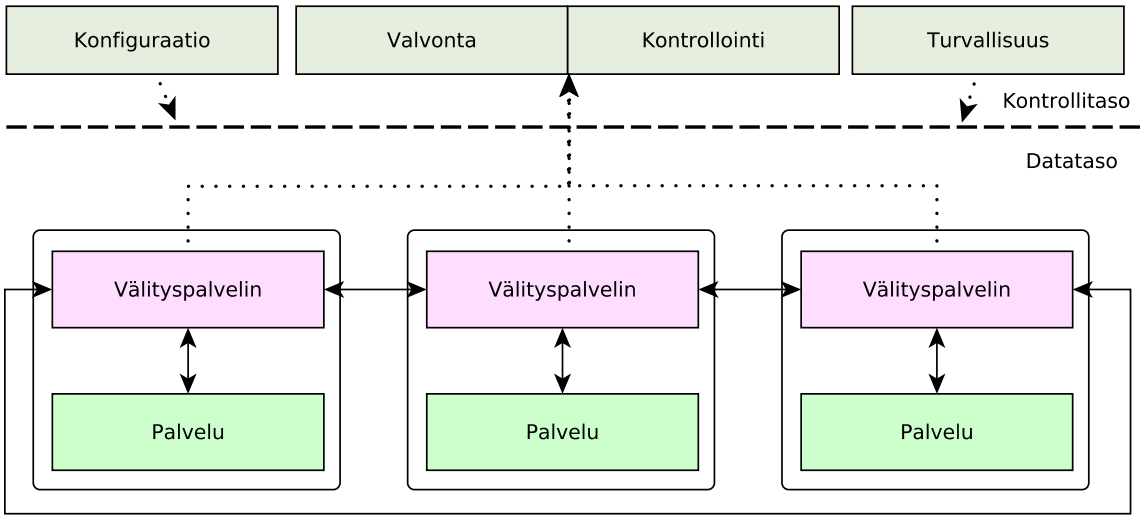
Taulukko 4.1: Kubernetes-termien kuvaukset.

4.2.2 Palveluverkot

Palveluverkolla (*service mesh*) tarkoitetaan infrastruktuurin kerrosta, jossa hallinnoidaan palveluiden välistä liikennettä (Morgan, 2017). Palveluverkkoja käytetään esimerkiksi pilviympäristöissä, joissa palvelut on kontitettu ja joissa kontitettuja palveluita hallinnoidaan konttien hallinnointijärjestelmällä. Kontitetussa ympäristössä yksi palvelu voi koostua useista palvelun instansseista (Morgan, 2017).

Palveluverkon voidaan ajatella koostuvan kahdesta tasosta, joita ovat *datataso*

(*data plane*) ja *kontrollitaso* (*control plane*) (Li, Lemieux, Gao, Zhao, & Han, 2019). Palveluverkon datataso koostuu palveluiden instanssien yhteyteen asennetuista välityspalvelimista, jotka välittävät liikennettä palveluiden instanssien välillä (Li et al., 2019). Palveluverkon kontrollitaso koostuu komponenteista, joiden avulla konfiguroidaan, valvotaan ja kontrolloidaan datatason komponentteja ja huolehditaan datatason komponenttien turvallisuudesta (Li et al., 2019). Palveluverkon yleinen arkkitehtuuri esitellään kuvassa 4.2.



Kuva 4.2: Palveluverkon arkkitehtuuri (Li et al., 2019).

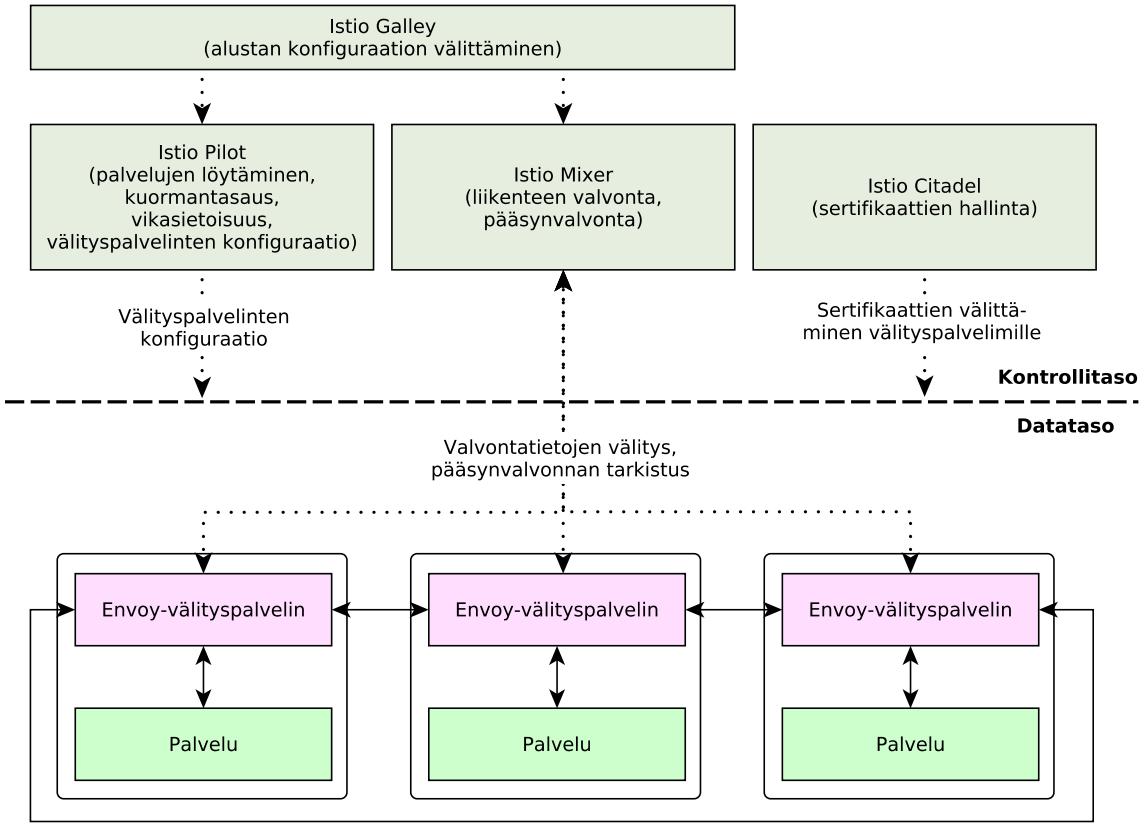
Ominaisuuksia, joita palveluverkot tarjoavat, ovat muun muassa *liikenteen kuormantasaus* (*load balancing*) ja *liikenteen valvonta* (*traffic monitoring*), *palvelujen löytäminen* (*service discovery*) sekä *autentikaatio ja pääsynvalvonta* (*authentication and access control*) (Li et al., 2019). Palveluverkoissa voidaan ottaa huomioon myös vikasietoisuus ohjaamalla palveluiden instanssien välistä liikennettä toimiviin ja kuormittamattomiin palveluiden instansseihin kuormitettujen tai toimimattomien instanssien sijaan (Li et al., 2019).

Yksi olemassa olevista palveluverkkoratkaisuista on Istio, joka on asennettavissa muun muassa Kubernetes-klusteriin ja joka tarjoaa mahdollisuuden ajonaikaisen liikenteenohjauksen toteuttamiseen (Istio Authors, 2019i). Kuvassa 4.3 esitellään Istion arkkitehtuuri ja taulukossa 4.2 esitellään yleisten Istio-termien kuvaukset.

Istion arkkitehtuuri koostuu kontrollitasosta ja datatasosta (Istio Authors, 2019a).

Istion datataso koostuu Envoy-välityspalvelimista[5], jotka välittävät liikennettä palveluiden välillä välityspalvelinten konfiguraatioiden mukaisesti. Envoy-välityspalvelimet asennetaan sivuvaunuina palveluiden yhteyteen.

Istion kontrollitaso koostuu Istion komponenteista, joita ovat Mixer, Pilot, Citadel ja Galley (Istio Authors, 2019a). Mixer-komponentin avulla kerätään Envoy-välityspalvelimilta liikenteeseen liittyviä valvontatietoja ja suoritetaan pääsynvalvontaa. Pilot-komponentilla välitetään palveluiden löytämiseen, kuormantasaukseen ja vikasietoisuuteen liittyviä konfiguraatiotietoja Envoy-välityspalvelimille. Citadel-komponentin avulla suoritetaan sertifikaattien hallintaa ja välitetään sertifikaatteja Envoy-välityspalvelimille. Galley-komponentin avulla eristetään Istion muut komponentit Kubernetes-alustasta ja välitetään alustan konfiguraatiotietoja Istion komponenteille.



Kuva 4.3: Istion arkkitehtuuri(Istio Authors, 2019a).

[5]https://www.envoyproxy.io/ (Vierailtu 27.12.2019).

Istio-termi	Kuvaus
Yhdyskäytävä (*Gateway*)	Yhdyskäytävällä tarkoitetaan kuormanjakajaa, joka ottaa vastaan saapuvia ja lähteviä verkkoyhteyksiä ja johon voidaan määritellä muun muassa palveluverkossa käytössä olevat portit (Istio Authors, 2019c).
VirtualService	VirtualService-konfiguraation avulla määritellään liikenteen ohjauksessa käytettävät säännöt (Istio Authors, 2019h).
DestinationRule	DestinationRule-konfiguraation avulla voidaan määritellä muun muassa konfiguraatiot kuormantasausta varten (Istio Authors, 2019b). Lisäksi DestinationRule-konfiguraatiossa voidaan määritellä *osajoukot* (*subset*), joihin esimerkiksi palvelun eri ohjelmistoversiot voidaan jakaa.

Taulukko 4.2: Istio-termien kuvaukset.

4.3 Kokeilun automatisointi

Tuotannossa kokeilua voidaan mallintaa muodollisella mallilla, jonka pohjalta voidaan toteuttaa menetelmä tuotannossa kokeilun automatisoimiseksi (Schermann, Schöni, Leitner, & Gall, 2016). Muodollinen malli perustuu viiteen vaatimukseen, jotka mallin toteuttava menetelmä täyttää. Taulukossa 4.3 esitellään muodollisen mallin vaatimukset ja vaatimusten kuvaukset.

Tuotannossa kokeilun muodollisen mallin toteuttavan Bifrost-prototyyppiohjelmiston avulla voidaan tuotannossa kokeilu automatisoida (Schermann et al., 2016). Bifrost-ohjelmiston arkkitehtuuri esitellään kuvassa 4.4. Bifrost-ohjelmisto muodostuu Bifrost-moottorista ja -välityspalvelimista. Bifrost-välityspalvelinten avulla käyttäjien liikennettä ohjataan ajonaikaisesti tuotannossa kokeilun aikana. Bifrost-moottorilla seurataan valvonnan kautta saatavia tietoja. Lisäksi Bifrost-moottorin avulla päivitetään Bifrost-välityspalvelinten käyttämiä liikenteenohjaussääntöjä tuotannossa kokeilun vaiheiden mukaan. Bifrost-välityspalvelinten käyttämiin liikenteenohjaussääntöihin tehdään muutoksia valvonnan kautta saatavien tietojen perusteella. Tuotannossa kokeilun etenemistä voidaan seurata Bifrost-tilannekuvanäkymän avulla. Tuotannossa kokeilun suoritusta voidaan hallita automatisoidusti Bifrost-komentorivityökalun avulla.

Sekä Kuberneteksen että Istion avulla on mahdollista toteuttaa tuotannossa kokeilua kandidaattiversion tai asteittaisen julkistamisen käytäntöä hyödyntäen. Kubernetes-klusterissa voidaan ohjelmistosta viedä tuotantoon käyttöön useita ver-

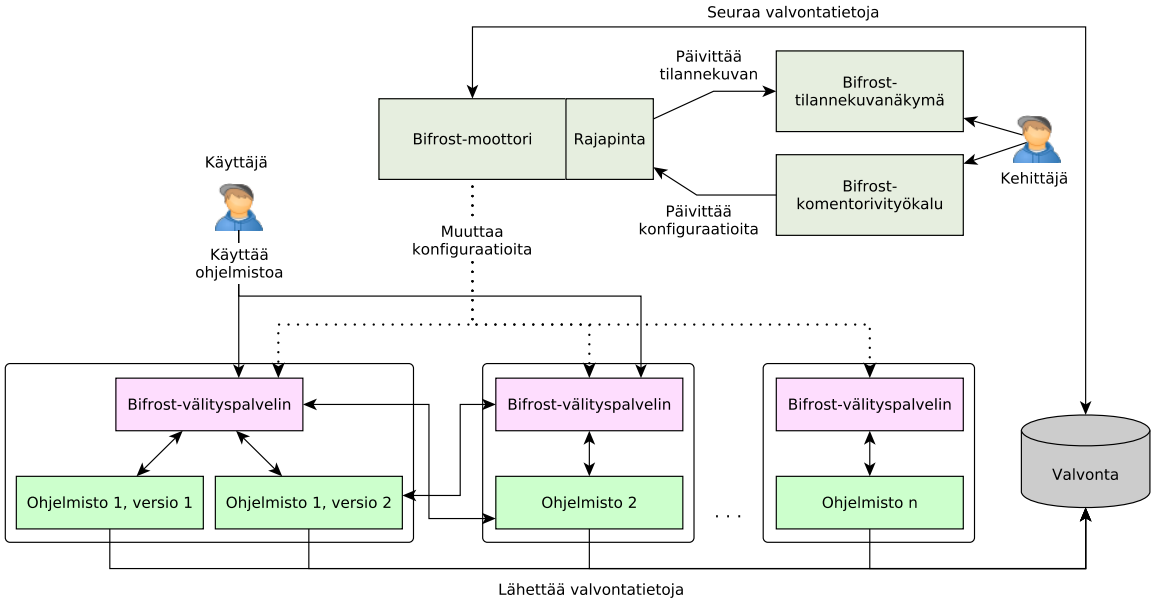
Ominaisuus	**Kuvaus**
Datalähtöisyys (*data-driven*)	Valvonnan kautta saatavien tietojen avulla voidaan ajonaikaisesti seurata järjestelmän tilaa tai päättää, oliko tuotannossa tehdyn kokeilun lopputulos onnistunut. Valvonnan kautta saatavia tietoja tulee siis voida hyödyntää tuotannossa kokeilun aikana.
Ajastettavuus (*timed execution*)	Tuotannossa kokeilun aikaiset tapahtumat, esimerkiksi käyttäjien ohjaaminen tiettyyn ohjelmistoversioon, tulee voida ajastaa tapahtuvaksi tietyllä hetkellä tuotannossa kokeilun aikana.
Yhtäaikaisuus (*parallel execution*)	Tuotannossa kokeilun aikana kokeiltavasta ohjelmistosta on voitava olla tuotannossa käytössä yhtä aikaa useita ohjelmistoversioita.
Liikenteen ohjattavuus (*traffic routing*)	Ohjelmiston käyttäjät on voitava ohjata käyttämään tiettyä ohjelmistoversiota tuotannossa kokeilun aikana.
Järjestettävyys (*ordered execution*)	Tuotanossa kokeilua voidaan tehdä käyttäen useaa käytäntöä, kuten kandidaattiversiota tai asteittaista julkistamista. Käytäntöjä tulee voida hyödyntää määritetyssä järjestyksessä tuotannossa kokeilun aikana.

Taulukko 4.3: Vaatimukset, joihin tuotannossa kokeilun muodollinen malli perustuu (Schermann et al., 2016).

sioita yhtäaikaisesti (The Kubernetes Authors, 2019d). Tällöin ohjelmistosta käytössä olevien Podien määrän avulla voidaan hallita, kuinka paljon liikennettä ohjelmistoversioihin ohjautuu (The Kubernetes Authors, 2019d). Jos ohjelmiston versiosta A on Kubernetes-klusterissa käytössä esimerkiksi yhdeksän Podia ja versiosta B on käytössä yksi Podi, ohjautuu yhdeksänkymmentä prosenttia liikenteestä ohjelmiston versioon A ja kymmenen prosenttia liikenteestä versioon B.

Käytettäessä Istiota tuotannossa kokeilun toteuttamiseen voidaan liikennettä ohjata ohjelmiston eri versioihin riippumatta siitä, kuinka paljon ohjelmistoversiosta on käytössä Podeja (Budinsky, 2018). Istion avulla voidaan VirtualService-konfiguraatiolla määritellä, kuinka monta prosenttia liikenteestä ohjataan ohjelmiston tiettyyn versioon. Istion avulla voidaan lisäksi ohjata liikennettä esimerkiksi HTTP-pyyntöjen otsakkeiden arvojen perusteella (Istio Authors, 2019e).

Kuberneteksen ja Istion tekemää liikenteenohjausta voidaan muuttaa päivittämällä Kuberneteksen ja Istion konfiguraatioita. Automatisoimalla Kuberneteksen ja Istion konfiguraatioiden päivitys voidaan Kubernetesta tai Istiota käyttää myös osana automatisoitua tuotannossa kokeilua.



Kuva 4.4: Bifrost-ohjelmiston arkkitehtuuri (Schermann et al., 2016).

5 Ohjelmiston vaiheittainen käyttöönotto kohdeyrityksessä

Tämä tutkielma on toteutettu keskisuurelle suomalaiselle ohjelmistopalveluita tarjoavalle yritykselle. Kohdeyritys toimii yritykseltä-yritykselle-toimialalla eli yrityksen tarjoamat ohjelmistopalvelut toteutetaan yrityksen asiakkaille. Seuraavissa aliluvuissa esitellään tutkimuksen tutkimusmenetelmä ja tutkimuksen aikana toteutetut sovellukset.

5.1 Tutkimusmenetelmä

Tässä tutkielmassa tutkimusmenetelmänä käytetään design science -tutkimusmenetelmää (Hevner et al., 2004). Design science -tutkimusmenetelmän avulla ratkaistaan organisaatiollisia ongelmia toteuttamalla ja arvioimalla tietoteknisiä artefakteja (Hevner et al., 2004). Design science -tutkimusmenetelmän lopputuotteena toimii artefakti, joka voi olla muun muassa malli, metodi tai ilmentymä esimerkiksi ohjelmiston muodossa (Hevner et al., 2004). Tässä tutkielmassa toteutettava artefakti on sovellus web-ohjelmiston vaiheittaisen käyttöönoton toteuttamiseksi Kubernetes-klusterissa.

Design science -tutkimusmenetelmä voidaan määritellä seitsemällä *suuntaviivalla* (*guideline*), joita voidaan käyttää apuna desing science -menetelmän vaatimusten

tunnistamisessa (Hevner et al., 2004). Suuntaviivat on esitelty taulukossa 5.1. Käytettäessä design science -tutkimusmenetelmää tulisi tehdyn tutkimuksen sisältää suuntaviivojen kuvaamat ominaisuudet (Hevner et al., 2004).

Suuntaviivassa numero 3 mainittuina arviointimenetelminä voivat toimia useat erilaiset tavat arvioida artefaktia (Hevner et al., 2004). Yksi esimerkki arviointimenetelmästä on simuloida artefaktin toimintaa keinotekoisella syötteellä. Toinen vaihtoehto on arvioida artefaktin hyödyllisyyttä olemassa olevan tieteellisen tutkimuksen perusteella.

Tavoitteena tutkimuksen esittelemisessä sekä tekniselle että hallinnolliselle kohdeyleisölle on, että kumpikin kohdeyleisö pystyy hyödyntämään tutkimuksessa kehitettyä artefaktia tarpeidensa mukaan (Hevner et al., 2004). Tekniselle kohdeyleisölle on tärkeää esitellä riittävästi tietoja, jotta artefakti on mahdollista tarvittaessa toteuttaa. Lisäksi tutkimuksen esittely mahdollistaa tutkimuksen toistettavuuden. Esiteltäessä tutkimusta hallinnolliselle kohdeyleisölle on tärkeää tarjota riittävästi tietoja artefaktin käyttöönottoon liittyvien päätösten tekemiseksi.

Design science -tutkimusmenetelmän avulla tehtävän tutkimuksen toteuttamiseksi on myös kehitetty metodologia, jossa hyödynnetään myös edellä mainittuja design science -tutkimusmenetelmän suuntaviivoja (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). Metodologia koostuu kuudesta vaiheesta (Peffers et al., 2007):

I Tutkimusongelman tunnistaminen ja motivaatio tutkimukselle ja perustelut tutkimusongelman ratkaisun tuomalle arvolle.

II Tavoitteiden määritteleminen ratkaisua varten ottaen huomioon mahdolliset olemassa olevat ongelmat ja ratkaisut sekä niiden tehokkuus.

III Artefaktin toiminnallisuuden ja arkkitehtuurin suunnittelu ja artefaktin toteuttaminen teoriatiedon pohjalta.

IV Tutkimusongelman ratkaiseminen artefaktin avulla esimerkiksi simulaation avulla.

V Artefaktin toiminnan arviointi mittaamalla ja tarkkailemalla artefaktin toimintaa verrattuna artefaktille asetettuihin tavoitteisiin.

VI Tutkimusongelman ja artefaktin esittely tutkimuksen kohdeyleisöille kertoen muun muassa tutkimusongelman merkityksestä ja artefaktin hyödyllisyydestä.

#	**Suuntaviiva**	**Kuvaus**
1	Artefaktin tuottaminen	Tutkimuksen tuloksena on tuotettava artefakti esimerkiksi mallina, metodina tai ilmentymänä.
2	Tutkimusongelman merkityksellisyys	Tutkimuksen tavoitteena on ratkaista liiketoimintaongelmia teknologiapohjaisten ratkaisujen avulla.
3	Mallin arviointi	Artefaktin hyödyllisyys, laatu ja tehokkuus tulee arviointimenetelmien avulla osoittaa täsmällisesti artefaktin teknisessä toimintaympäristössä.
4	Tieteellinen kontribuutio	Tutkimuksen tulee tuottaa lopputuloksena tieteellisiä kontribuutioita esimerkiksi artefaktin muodossa.
5	Tutkimuksen täsmällisyys	Artefaktin toteuttamisessa ja arvioimisessa tulee käyttää täsmällisiä tutkimusmenetelmiä ja esimerkiksi tietoja, jotka pohjautuvat olemassa olevaan tutkimukseen.
6	Tutkimus etsivänä prosessina	Tavoittena on kehittää artefakti iteratiivisesti käytössä olevia keinoja hyväksikäyttäen samalla ottaen huomioon ongelma-alueen organisaatiolliset ja tekniset rajoitukset ja vaatimukset.
7	Tutkimuksen esitteleminen	Tutkimus tulee esitellä ymmärrettävästi sekä tekniselle että hallinnolliselle kohdeyleisölle.

Taulukko 5.1: Design science -tutkimusmenetelmän suuntaviivat (Hevner et al., 2004).

Metodologiassa vaiheen V jälkeen voidaan päättää, siirrytäänkö takaisin vaiheeseen III, jotta artefaktin toimintaa voidaan parantaa (Peffers et al., 2007). Vaiheiden III, IV ja V toistaminen mahdollistaa artefaktin toiminnallisuuden kehittämisen iteratiivisesti. Toinen vaihtoehto on jättää artefaktin jatkokehitys tuleviin projekteihin ja siirtyä vaiheeseen VI (Peffers et al., 2007). Metodologian vaiheista huomataan, että ne sisältävät design science -tutkimusmenetelmän suuntaviivoja. Vaiheet I ja II sisältävät suuntaviivan numero 2. Vaiheet III, IV ja V sisältävät kokonaisuutena suuntaviivat 1, 3, 5 ja 6. Viimeinen vaihe numero VI sisältää suuntaviivat 4 ja 7.

5.2 Vaatimukset ja tavoitteet

Tutkielman lähtökohtana on kohdeyrityksen toive luoda pohjaa jatkuvalle kokeilulle. Yksi tapa luoda pohjaa jatkuvalle kokeilulle on kehittää käytäntöjä ohjelmistojen vaiheittaiseen käyttöönottoon. Kohdeyrityksessä halutaan kehittää vaiheittaisen käyttöönoton käytäntöjä Kubernetes-ympäristöön, jossa vaatimuksena on ympäristössä

toimivien ohjelmistojen käytön katkottomuus.

Tässä tutkielmassa rajoitutaan tarkastelemaan vaiheittaista käyttöönottoa tilattomien web-ohjelmistojen osalta. Jatkuvan ohjelmistotuotannon käytännöistä kohdeyrityksessä on käytössä jatkuva integraatio. Lisäksi tuotantoon viennit tehdään jatkuvan toimittamisen avulla, sillä uuden ohjelmistoversion vieminen tuotantoon halutaan käynnistää manuaalisesti, kun päätös uuden ohjelmistoversion tuotantoon viemisestä tehdään.

Taulukossa 5.2 esitellään otsikkotasolla vaatimukset, joita odotetaan uuden ohjelmistoversion vaiheittaisen käyttöönoton menetelmältä. Vaatimukset on ryhmitelty kolmeen ryhmään, joita ovat tekniset vaatimukset, vaiheittaisen käyttöönoton vaatimukset ja vaiheittaisen käyttöönoton päättymiseen liittyvät vaatimukset. Tarkemmat kuvaukset vaatimuksista esitellään tämän aliluvun alakohdissa.

Vaatimuksen numero	**Vaatimuksen ryhmä**	**Vaatimus**
1	Tekniset vaatimukset	Ohjelmiston lähdekoodin muuttumattomuus
2	Tekniset vaatimukset	Ohjelmistoversioiden rinnakkaisuus
3	Tekniset vaatimukset	Käyttökatkon pituus
4	Vaiheittaisen käyttöönoton vaatimukset	Ohjelmiston käyttäjäryhmän rajoittaminen
5	Vaiheittaisen käyttöönoton vaatimukset	Vaiheittaisen käyttöönoton kesto
6	Vaiheittaisen käyttöönoton vaatimukset	Käyttäjämäärä vaiheittaisen käyttöönoton aikana
7	Vaiheittaisen käyttöönoton päättyminen	Vanhan ohjelmistoversion poistaminen
8	Vaiheittaisen käyttöönoton päättyminen	Vaiheittaisen käyttöönoton keskeyttäminen

Taulukko 5.2: Vaatimukset vaiheittaisen käyttöönoton menetelmälle.

5.2.1 Tekniset vaatimukset

Teknisillä vaatimuksilla halutaan varmistaa, että uuden ohjelmistoversion vaiheittaisen käyttöönoton menetelmä toteuttaa tarvittavat tekniset ominaisuudet.

Vaatimus 1: Ohjelmiston lähdekoodin muuttumattomuus

Kuvaus: Uuden ohjelmistoversion vaiheittainen käyttöönotto on pystyttävä toteuttamaan sellaisella menetelmällä, joka ei vaadi muutoksia käyttöönotettavan ohjelmiston lähdekoodiin.

Perustelut: Tavoite on, että lähdekoodi ei ole riippuvainen käytetystä vaiheittaisen käyttöönoton menetelmästä.

Vaatimus 2: Ohjelmistoversioiden rinnakkaisuus

Kuvaus: Ohjelmistosta on voitava ottaa tuotantoon käyttöön muokattavissa olevaksi ajaksi uusi ohjelmistoversio vanhan ohjelmistoversion rinnalle. Muokattavissa olevan ajan tulee voida vaihdella minuuttien ja päivien välillä.

Perustelut: Uuden ja vanhan ohjelmistoversion tulee voida olla rinnakkain käytössä tuotannossa, jotta uutta ohjelmistoversiota voidaan testata tietyn ajan verran vanhan ohjelmistoversion rinnalla.

Vaatimus 3: Käyttökatkon pituus

Kuvaus: Ohjelmiston uuden version vaiheittaisen käyttöönoton aikana ohjelmiston käyttökatkon tulee olla mahdollisimman lyhyt.

Perustelut: Tavoitteena on, ettei ohjelmiston käytössä ole käyttökatkoa uuden ohjelmistoversion vaiheittaisen käyttöönoton aikana. Hyväksyttävä yläraja käyttökatkon pituudelle on kymmenen sekuntia. Mahdollisimman lyhyt käyttökatko halutaan, jotta uuden ohjelmistoversion vaiheittaisesta käyttöönotosta ei tule haittaa ohjelmiston käyttäjille.

5.2.2 Vaiheittaisen käyttöönoton vaatimukset

Vaiheittaisen käyttöönoton vaatimuksilla kuvataan, miten vaiheittaisen käyttöönoton tulee edetä.

Vaatimus 4: Ohjelmiston käyttäjäryhmän rajoittaminen

Kuvaus: Uuteen ohjelmistoversioon tulee voida antaa pääsy määrätylle käyttäjäryhmälle. Määrätty käyttäjäryhmä voi olla yksi tai useampi yksittäinen käyttäjä tai organisaatio.

Perustelut: Määrätyn käyttäjäryhmän jäsenten pääsy uuteen ohjelmistoversioon halutaan, jotta uutta ohjelmistoversiota voidaan testata määrätyllä käyttäryhmällä ja kerätä testauksen päättyessä käyttäjäryhmän jäseniltä palautetta uudesta ohjelmistoversiosta.

Vaatimus 5: Vaiheittaisen käyttöönoton kesto

Kuvaus: Vaiheittainen käyttöönotto tulee voida suorittaa loppuun asti muokattavissa olevan ajan sisällä. Muokattavissa olevan ajan tulee voida vaihdella minuuttien ja päivien välillä.

Perustelut: Vaiheittaisen käyttöönoton keston muokattavissa oleva aika halutaan, jotta vaiheittaisen käyttöönoton kestoa voidaan muuttaa tarpeen mukaan pidemmäksi tai lyhyemmäksi.

Vaatimus 6: Käyttäjämäärä vaiheittaisen käyttöönoton aikana

Kuvaus: Vaiheittainen käyttöönotto tulee voida suorittaa loppuun asti muokattavissa olevan kokoisissa vaiheissa. Muokattavan kokoiset vaiheet tulee voida määrittää prosenttiosuutena käyttäjistä, joilla ei vielä ole pääsyä uuteen ohjelmistoversioon, mutta joilla on oikeus käyttää ohjelmistoa.

Perustelut: Muokattavan kokoiset vaiheet tarvitaan, jotta uuden ohjelmistoversion kuormaa voidaan vaiheittain lisätä ja seurata, aiheutuuko kuorman lisäämisestä ongelmia.

5.2.3 Vaiheittaisen käyttöönoton päättyminen

Vaiheittaisen käyttöönoton päättymiseen liittyvillä vaatimuksilla kuvataan, miten vaiheittaisen käyttöönoton tulee päättyä käyttöönoton onnistuessa tai virhetilanteessa.

Vaatimus 7: Vanhan ohjelmistoversion poistaminen

Kuvaus: Vanhan ohjelmistoversion tulee poistua automaattisesti käytöstä, kun uusi ohjelmistoversio on käytössä kaikilla käyttäjillä, joilla on oikeus käyttää ohjelmistoa.

Perustelut: Vanha ohjelmistoversio tulee poistua käytöstä, jotta vanha ohjelmistoversio ei enää kuluta järjestelmän resursseja.

Vaatimus 8: Vaiheittaisen käyttöönoton keskeyttäminen

Kuvaus: Ohjelmiston uuden version vaiheittainen käyttöönotto tulee voida keskeyttää ja palata takaisin vanhan ohjelmistoversion käyttöön. Keskeytettäessä vaiheittainen käyttöönotto uutta ohjelmistoversiota käyttäneiden käyttäjien tulee palata käyttämään vanhaa ohjelmistoversiota. Uuden ohjelmistoversion tulee poistua automaattisesti käytöstä järjestelmästä, kun ongelmia ohjelmiston toiminnassa havaitaan.

Perustelut: Vaiheittaisen käyttöönoton keskeytystä tarvitaan tilanteessa, jossa ohjelmiston uuden version vaiheittaisen käyttöönoton aikana ohjelmiston toiminnassa havaitaan ongelmia. Tavoitteena on, että ongelmatilanteissa ohjelmiston käyttäjille koitunut haitta on mahdollisimman pieni.

5.3 Vaiheittain käyttöönotettavan ohjelmiston esittely

Ohjelmiston vaiheittaisen käyttöönoton testaamista varten tarvitaan ohjelmisto, jota voidaan käyttää päivitettävänä ohjelmistona testitapauksissa. Tätä tarkoitusta varten on kehitetty Aforismi-sovellus. Aforismi-sovellus on *palvelinsovellus* (*backend*), joka tarjoaa rajapinnan, johon voidaan tehdä tietopyyntöjä HTTP-protokollan[6] avulla.

Aforismi-sovellus soveltuu uuden ohjelmistoversion vaiheittaisen käyttöönoton testaamiseen, sillä vaiheittaisen käyttöönoton etenemistä voidaan seurata koneellisesti Aforismi-sovelluksen tarjoaman rajapinnan kautta. Rajapinnan kautta tehtävä seuraaminen mahdollistaa rajapintaan tulevien pyyntöjen HTTP-statuskoodien[7] seuraamisen ja rajapinnan palauttamien vastausten näkemisen JSON-muodossa[8].

Palvelinsovelluksen tarjoama rajapinta on kuvattu taulukossa 5.3. Rajapinnassa tarjotaan polku yksittäisen aforismin tietojen hakemiseen palvelinsovellukselta. Yksittäisen aforismin tiedot palautetaan rajapinnasta JSON-oliona `/aphorism`-polusta. Aforismin tietoihin sisältyvät aforismin otsikko ja kuvaus. Lisäksi JSON-olion tiedoissa palautetaan rajapintaan tehdyn pyynnön mukana lähetettyjen valinnaisten parametrien tai otsakkeiden arvot. JSON-olion tietoihin lisätään myös palvelinsovelluksen

[6] https://tools.ietf.org/html/rfc7231 (Vierailtu 29.11.2019).
[7] https://tools.ietf.org/html/rfc7231 (Vierailtu 29.11.2019).
[8] https://tools.ietf.org/html/rfc8259 (Vierailtu 11.12.2019).

versionumero ja aikaleima hetkestä, jolloin rajapintapyyntö saapui palvelinsovelluksen käsiteltäväksi.

HTTP-metodi	GET
Rajapinnan polku	/aphorism
Rajapinnan parametrit	”user” (valinnainen tekstimuotoinen käyttäjä) ”organization” (valinnainen tekstimuotoinen organisaatio)
Rajapinnan otsakkeet	”Aphorism-User” (valinnainen tekstimuotoinen käyttäjä) ”Aphorism-Organization” (valinnainen tekstimuotoinen organisaatio)

Vastauksen JSON-olio	`{"title": "Otsikko",`
	`"description": "Kuvaus",`
	`"user": "Parametrina lähetetty käyttäjä",`
	`"organization": "Parametrina lähetetty organisaatio",`
	`"timestamp": "Aikaleima hetkestä, jolloin pyyntö saapui palvelinsovellukseen",`
	`"headerUser": "Otsakkeen arvona lähetetty käyttäjä",`
	`"headerOrganization": "Otsakkeen arvona lähetetty organisaatio",`
	`"backendVersion": "Palvelinsovelluksen versionumero"}`

Taulukko 5.3: Aforismi-sovelluksen tarjoama rajapinta.

Aforismi-sovelluksen toimintaa voidaan kuvata kahden käyttötapauksen avulla. Käyttötapaukset on määritelty sekä tilanteelle, jossa aforismin hakeminen onnistuu, että tilanteelle, jossa aforismin hakeminen epäonnistuu. Aforismi-sovelluksen avulla on siis mahdollista simuloida myös tilannetta, jossa uusi ohjelmistoversio on virheellinen. Taulukosta 5.4 löytyvät versiokohtaisesti vaihtelevat tiedot, jotka Aforismi-sovelluksen tarjoama rajapinta palauttaa JSON-olion mukana. Lisäksi taulukon sarakkeeseen ”Palautetaan virhe” on kirjattu ehto, jonka tulee täyttyä, jotta aforismin hakeminen epäonnistuu.

Versionumero	**Otsikko**	**Kuvaus**	**Palautetaan virhe**
1.7.0	Elä niin kuin kuolisit huomenna. Opi niin kuin eläisit ikuisesti. - Mahatma Gandhi	Oppiminen ja eläminen	-
1.8.0	Kyllä se siitä - Motivaatiovalas	Motivoiva toteamus	-
1.9.0	Kyllä se siitä - Motivaatiovalas	Motivoiva toteamus	Otsakkeen ”Aphorism-Organization” sisältäessä merkit ”apho-deploy”.
1.9.1	Kyllä se siitä - Motivaatiovalas	Motivoiva toteamus	Otsakkeen ”Aphorism-Organization” ensimmäisen merkin ollessa ”A”.

Taulukko 5.4: Aforismi-sovelluksen palauttamat tiedot versioittain.

Käyttötapaus 1: Aforismin hakeminen onnistuu

Aforismi-sovelluksen versiot: 1.7.0 ja 1.8.0

Aforismi-sovelluksen toiminta: Aforismi-sovelluksen tarjoama rajapinta palauttaa JSON-olion ilman virheitä. JSON-oliossa palautettavat versioittain vaihtelevat tiedot löytyvät taulukosta 5.4 versioiden 1.7.0 ja 1.8.0 kohdalta.

Käyttötapaus 2: Aforismin hakeminen epäonnistuu

Aforismi-sovelluksen versiot: 1.9.0 ja 1.9.1

Aforismi-sovelluksen toiminta: Aforismi-sovelluksen tarjoama rajapinta palauttaa JSON-olion ilman virheitä, jos virheen ehto ei täyty. JSON-oliossa palautettavat versioittain vaihtelevat tiedot löytyvät taulukosta 5.4 versioiden 1.9.0 ja 1.9.1 kohdalta. Jos virheen ehto täyttyy, Aforismi-sovelluksen tarjoama rajapinta palauttaa poikkeuksen ”AphorismException”. Poikkeuksen lisäksi rajapinta palauttaa virhekoodin ”500 Internal Server Error”[9].

[9]https://tools.ietf.org/html/rfc7231 (Vierailtu 29.11.2019).

5.4 Vaiheittaisen käyttöönoton menetelmän esittely

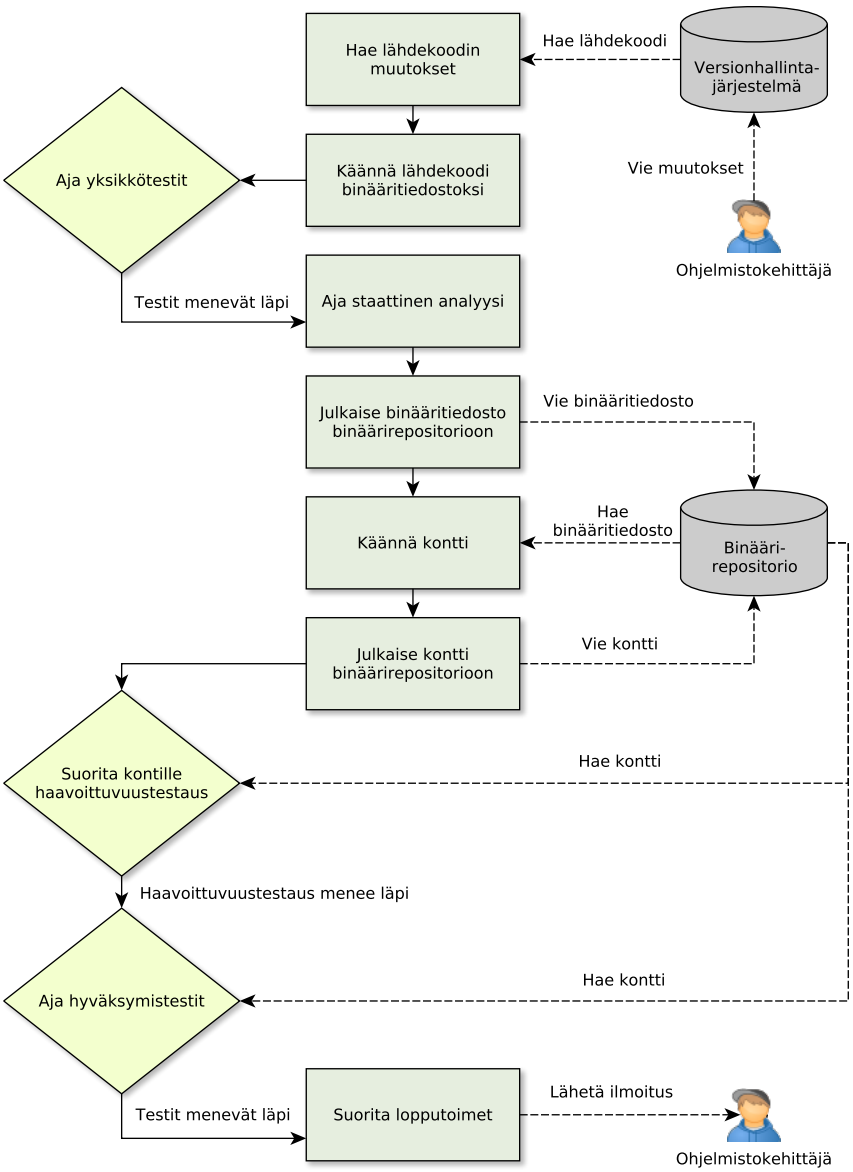
Kuvassa 5.1 esitellään kohdeyrityksessä käytössä oleva tuotantoon viennin putki, jonka päivitettävä ohjelmisto käy läpi ennen siirtymistä julkaisuvaiheeseen. Tuotantoon viennin putki käynnistyy, kun ohjelmistokehittäjä vie versionhallintajärjestelmään uuden muutoksen. Putki alkaa lähdekoodin muutosten hakemisella versionhallintajärjestelmästä, minkä jälkeen ohjelmistosta käännetään suoritettava ohjelmistopaketti. Ohjelmistopaketille suoritetaan yksikkötestit. Yksikkötestien onnistuttua ohjelmistokoodille tehdään staattinen analyysi, jonka jälkeen käännetty ajettava ohjelmistopaketti viedään *binäärirepositorioon* (*binary repository*).

Seuraavassa vaiheessa ohjelmistopaketista luodaan kontti, joka julkaistaan binäärirepositorioon. Kontille suoritetaan haavoittuvuustestaus, jonka onnistuneen suorituksen jälkeen ohjelmisto testataan hyväksymistestauksessa. Hyväksymistestauksen testien onnistuneen suorituksen päätyttyä ohjelmistokehittäjälle lähetetään tieto tuotantoon viennin putken päättymisestä.

Tässä tutkielmassa täydennetään olemassa olevaa tuotantoon viennin putkea kehittämällä menetelmä web-ohjelmiston vaiheittaiseen käyttöönottoon. Vaiheittaisen käyttöönoton menetelmä voisi toimia seuraavana askeleena kuvassa 5.1 esitellyn tuotantoon viennin putken lopputoimien jälkeen.

Luvussa 4 esiteltiin menetelmä, jonka avulla tuotannossa suoritettava kokeilu voidaan automatisoida. Menetelmässä hyödynnetään välityspalvelimia, joiden avulla liikenteelle voidaan suorittaa ajonaikaista liikenteenohjausta (Schermann et al., 2016). Käytettäessä ajonaikaista liikenteenohjausta tuotannossa suoritettavan kokeilun menetelmänä tuotannossa kokeiltavan ohjelmiston lähdekoodiin ei ole tarvetta tehdä muutoksia, sillä ajonaikainen liikenteenohjaus tehdään välityspalvelinten avulla (Schermann et al., 2016).

Tässä tutkielmassa vaiheittaisen käyttöönoton automatisoimiseksi toteutettiin sovellus AphoDeploy. Lisäksi ajonaikaisen liikenteenohjauksen toteutuksessa hyödynnettiin palveluverkkoratkaisu Istiota. AphoDeploy-sovelluksen ja Istion käytön yhdistelmä vastaa luvun 4 kuvassa 4.4 esiteltyä tekniikkaa käyttää välityspalvelimia ajonaikaiseen liikenteenohjaukseen ja erillistä sovellusta ajonaikaisen liikenteenohjauksen konfiguraatioiden muuttamiseen.



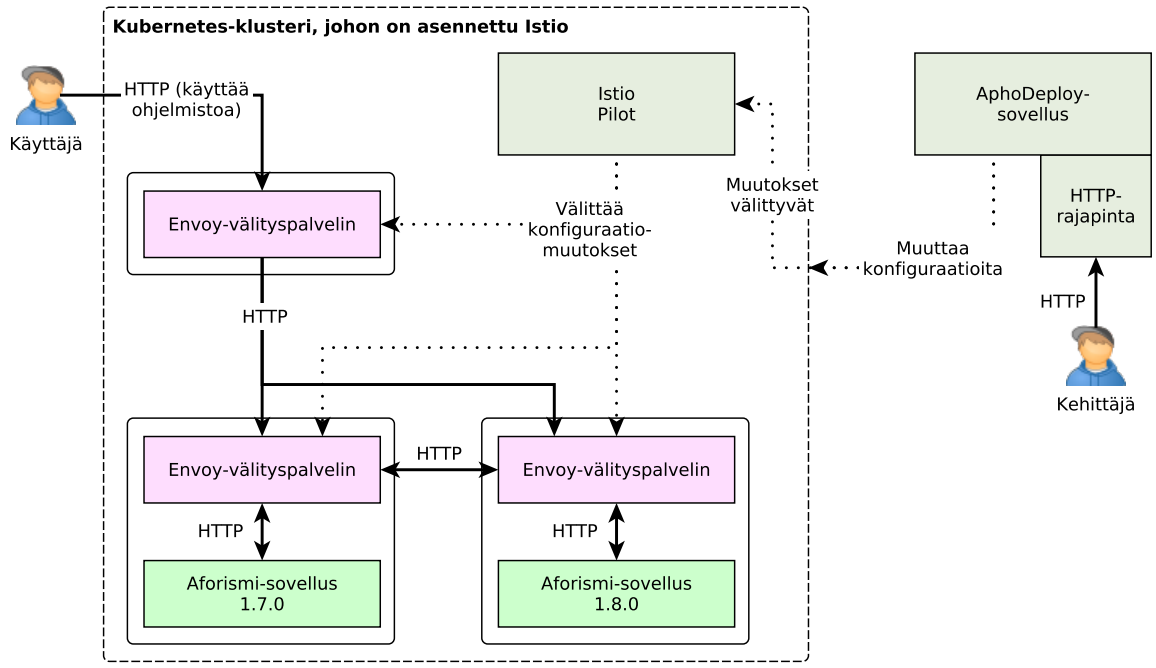
Kuva 5.1: Käytössä oleva tuotantoon viennin putki.

Kuvassa 5.2 esitellään tutkielmassa toteutetun kokonaisuuden korkean tason arkkitehtuuri. Kuvasta nähdään, että AphoDeploy-sovelluksen avulla Istion konfiguraatioihin voidaan tehdä muutoksia, jotka Istion Pilot-palvelu[10] välittää Istion tarjoamille Envoy-välityspalvelimille. Envoy-välityspalvelimet ohjaavat ajonaikaisesti liikennettä konfiguraatioiden perusteella Aforismi-sovelluksen eri versioihin. AphoDeploy-sovelluksen avulla Istion konfiguraatioihin tehtävät muutokset voidaan tehdä ohjelmallisesti esimerkiksi tietyillä ajan hetkillä.

AphoDeploy-sovellus tarjoaa HTTP-rajapinnan, jonka avulla vaiheittainen käyt-

[10]https://istio.io/docs/ops/deployment/architecture/ (Vierailtu 11.12.2019).

töönotto voidaan käynnistää. AphoDeploy-sovelluksen tarjoama rajapinta esitellään taulukossa 5.5. AphoDeploy-sovelluksen testaamiseksi toteutettujen testitapausten kuvaukset esitellään luvussa 6. Rajapinnan kautta voidaan käynnistää AphoDeploy-sovellusta testaavien testitapausten mukaiset vaiheittaiset käyttöönotot tai Kubernetes-klusterin tilan palauttaminen testitapausten suoritusten jälkeen. Kaikki AphoDeploy-sovelluksen rajapinnan tarjoamat metodit käynnistävät metodia vastaavan testitapauksen tai tilan palauttamisen asynkronisesti ja palauttavat rajapinnan kutsujalle vastauksena heti onnistuneen vastauksen.

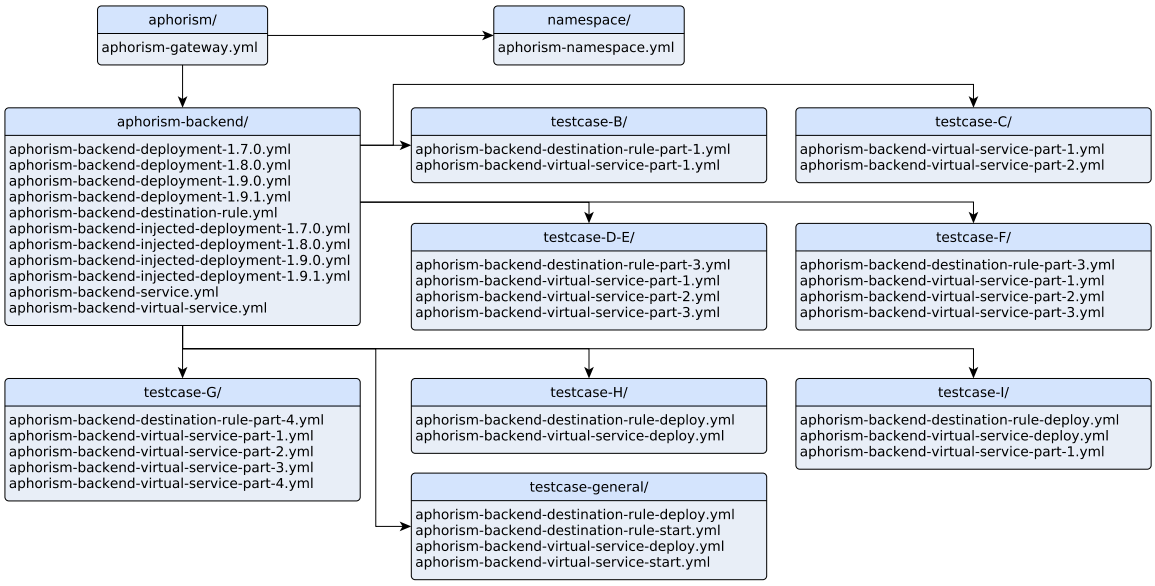


Kuva 5.2: Tutkielmassa toteutetun kokonaisuuden korkean tason arkkitehtuuri.

Palveluverkkoratkaisu Istio asennettiin *paikalliseen itse isännöityyn* (*on-premises*) Kubernetes-klusteriin. Istion asennuksen jälkeen Aforismi-sovellusta varten toteutettiin Istion vaatimat konfiguraatiotiedostot, joita käytettiin liikenteen ohjauksessa Istion avulla. Kuvassa 5.3 esitellään Aforismi-sovelluksen Kubernetes-konfiguraatioiden kansiorakenne. Aforismi-sovellus asennettiin Kubernetes-klusteriin omaan nimiavaruuteensa. Aforismi-sovelluksen palvelinsovellus asennettiin Kubernetes-klusteriin määrittelemällä Aforismi-sovellusta varten Kubernetes Deployment ja Service. Jokaiselle Aforismi-sovelluksen versiolle määriteltiin erillinen Deployment-konfiguraatio.

HTTP-metodi	Rajapinnan polku	Kuvaus
GET	/testcase-A	Käynnistää testitapaukseen A liittyvän vaiheittaisen käyttöönoton.
GET	/testcase-B	Käynnistää testitapaukseen B liittyvän vaiheittaisen käyttöönoton.
GET	/testcase-C	Käynnistää testitapaukseen C liittyvän vaiheittaisen käyttöönoton.
GET	/testcase-D	Käynnistää testitapaukseen D liittyvän vaiheittaisen käyttöönoton.
GET	/testcase-E	Käynnistää testitapaukseen E liittyvän vaiheittaisen käyttöönoton.
GET	/testcase-F	Käynnistää testitapaukseen F liittyvän vaiheittaisen käyttöönoton.
GET	/testcase-G	Käynnistää testitapaukseen G liittyvän vaiheittaisen käyttöönoton.
GET	/testcase-H	Käynnistää testitapaukseen H liittyvän vaiheittaisen käyttöönoton.
GET	/testcase-I	Käynnistää testitapaukseen I liittyvän vaiheittaisen käyttöönoton.
GET	/reset-testcase	Palauttaa Kubernetes-klusterin tilan testitapausten lähtötilanteeseen.

Taulukko 5.5: AphoDeploy-sovelluksen tarjoama rajapinta.



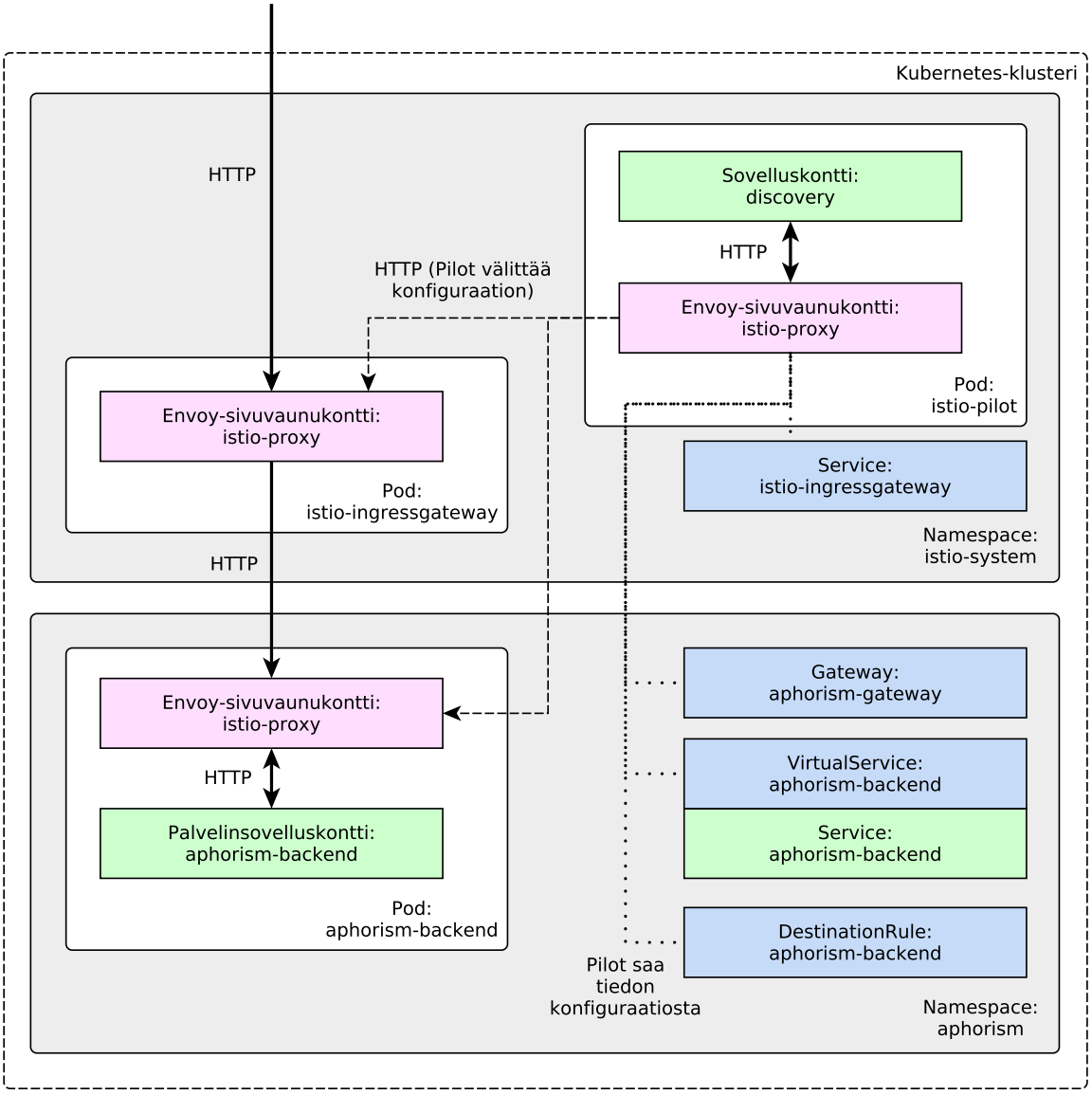
Kuva 5.3: Aforismi-sovelluksen Kubernetes-konfiguraatioiden kansiorakenne.

Istiota varten tavallisten Kubernetes-konfiguraatioiden lisäksi Aforismi-sovellukselle tehtiin Deployment-konfiguraatiot, joihin injektoitiin Istion Envoy-sivuvaunun konfiguraatio käyttäen manuaalista sivuvaunun injektointia[11]. Injektointi suoritettiin jokaiselle Aforismi-sovelluksen Deployment-konfiguraatiolle. Istiota varten luotiin myös Gateway-, VirtualService- ja DestinationRule-konfiguraatiot. Lisäksi jokaista luvussa 6 esiteltävää testitapausta varten luotiin konfiguraatiot, joiden avulla testitapauksissa kuvatut toiminnallisuudet on toteutettu.

Aforismi-sovelluksen Kubernetes-arkkitehtuuri Istion asennuksen ja konfiguroinnin jälkeen on esitelty kuvassa 5.4. Kuvasta nähdään, että liikenne Aforismi-sovellukselle kulkee Envoy-sivuvaunukonttien[12] kautta. Ensimmäiseksi liikenne vastaanotetaan nimiavaruudessa ”istio-system”, josta ”istio-ingressgateway”-Podissa sijaitseva Envoy-sivuvaunukontti ohjaa liikenteen nimiavaruudessa ”aphorism” sijaitsevaan Podiin ”aphorism-backend”. Podin ”aphorism-backend” sisällä sijaitseva Envoy-sivuvaunukontti ohjaa liikenteen Podin sisällä sijaitsevaan palvelinsovelluskonttiin ”aphorism-backend”. Envoy-sivuvaunukonteille välitetään liikenteen ohjaamista varten tarvitavat konfiguraatiot Istion Pilot-palvelun avulla. Liikenteenohjausta voidaan muuttaa muuttamalla Istion VirtualService- ja DestinationRule-konfiguraatioita.

AphoDeploy-sovelluksen toteutuksessa on otettu huomioon Istion käytäntö HTTP-virhekoodin ”503 Service Unavailable”[13] välttämiseksi suunniteltaessa järjestystä, jossa Istion konfiguraatioihin tehdään muutoksia. Istion käytännössä on mainittu, että DestinationRule-konfiguraation päivityksen jälkeen ennen VirtualService-konfiguraation päivitystä tulee odottaa muutama sekunti, jotta DestinationRule-konfiguraatioon tehty muutos ehtii päivittyä Envoy-sivuvaunuille (Istio Authors, 2019f).

[11] https://istio.io/docs/setup/additional-setup/sidecar-injection/ (Vierailtu 11.12.2019).
[12] https://istio.io/docs/ops/deployment/architecture/ (Vierailtu 11.12.2019).
[13] https://tools.ietf.org/html/rfc7231 (Vierailtu 29.11.2019).



Kuva 5.4: Aforismi-sovelluksen Kubernetes-arkkitehtuuri.

Maininta muutaman sekunnin odotusajasta ei kuitenkaan ole tarkka. Tämän vuoksi aloitettaessa vaiheittainen käyttöönotto tehdään liikenteenohjaus AphoDeploy-sovelluksen avulla kahdessa vaiheessa. Ensimmäisessä vaiheessa päivitetään DestinationRule-konfiguraatio ja ohjataan heti sen jälkeen AphoDeploy-sovelluksen liikenne uuteen versioon VirtualService-konfiguraation avulla. Muiden käyttäjien liikenne ohjataan edelleen vanhaan versioon. Tämän jälkeen siirrytään odotusaikaan, jonka aikana AphoDeploy-sovellus odottaa onnistunutta vastausta uudelta ohjelmistoversiolta.

Toisessa vaiheessa, kun AphoDeploy-sovellus on saanut uudelta ohjelmistover-

siolta onnistuneen vastauksen, päivitetään VirtualService-konfiguraatiota uudestaan ohjaamaan AphoDeploy-sovelluksen lisäksi myös muiden käyttäjien liikenne uuteen versioon. Tämän menetelmän avulla tiedetään tarkalleen, milloin muiden käyttäjien liikenne voidaan ohjata uuteen versioon. Menetelmän avulla varmistetaan myös, että käyttäjille ei aiheudu käyttökatkoa liikenteen ohjaukseen tehtävien muutosten yhteydessä.

AphoDeploy-sovelluksen toiminnan esittelemiseen voidaan käyttää luvussa 6 esiteltävää testitapausta B. Testitapauksessa B suoritetaan ensimmäisenä Aforismi-sovelluksen versiolle 1.8.0 vaiheittainen käyttöönotto, jonka jälkeen Aforismi-sovelluksen versio 1.7.0 poistetaan käytöstä. Testitapauksen suorituksen vaiheiden järjestys AphoDeploy-sovelluksessa on seuraavanlainen:

Lähtötilanne:

- Kubernetes-klusterissa on Aforismi-sovelluksen version 1.7.0 Podi.
- Istion avulla on ohjattu kaikki liikenne Aforismi-sovelluksen versioon 1.7.0.
- Aforismi-sovelluksen versio 1.8.0 on käännettynä binäärirepositoriossa.
- Aforismi-sovellukselle on toteutettu tarvittavat Kubernetes-konfiguraatiot ja Istio-konfiguraatiot testitapausta varten.

Vaiheittaisen käyttöönoton yleinen vaihe:

1. Konfiguroidaan Kubernetes Deployment Aforismi-sovelluksen versiolle 1.8.0.
2. Odotetaan, että version 1.8.0 Kubernetes Deployment Podit ovat valmiit.
3. Konfiguroidaan Istion DestinationRule, joka sisältää osajoukot Aforismi-sovelluksen versioille 1.7.0 ja 1.8.0.
4. Konfiguroidaan Istion VirtualService, joka ohjaa AphoDeploy-sovelluksen liikenteen versioon 1.8.0 ja muiden käyttäjien liikenteen versioon 1.7.0.
5. Odotetaan maksimissaan 120 sekuntia, että AphoDeploy-sovellus saa versiolta 1.8.0 onnistuneen vastauksen.

Vaiheittaisen käyttöönoton testitapauskohtainen vaihe:

1. Konfiguroidaan Istion VirtualService, joka ohjaa kaiken liikenteen Aforismi-sovelluksen versioon 1.8.0.
2. Konfiguroidaan Istion DestinationRule, joka sisältää osajoukon vain versiolle 1.8.0.
3. Poistetaan Aforismi-sovelluksen version 1.7.0 Kubernetes Deployment.
4. Odotetaan 60 sekuntia testitapauksen päättymistä.

Vaiheittaisen käyttöönoton testitapauskohtaisessa vaiheessa noudatetaan Istion käytäntöä osajoukon poistamisen kohdalla. Istion käytäntö on poistaa osajoukko ensin VirtualService-konfiguraatiosta, minkä jälkeen osajoukko voidaan poistaa DestinationRule-konfiguraatiosta (Istio Authors, 2019f). Testitapauksen B kuvauksesta huomataan, että Aforismi-sovelluksen versioon 1.7.0 ohjatun liikenteen poistaminen suoritetaan Istion käytännön mukaisessa järjestyksessä, ennen kuin version 1.7.0 Kubernetes Deployment poistetaan Kubernetes-klusterista.

6 Vaiheittaisen käyttöönoton menetelmän arviointi

Vaiheittaisen käyttöönoton menetelmää arvioidaan testitapauksilla, joiden avulla varmistetaan, toteuttaako vaiheittaisen käyttöönoton menetelmä sille edellä asetetut vaatimukset. Testitapausten sisältö ja niiden rakenne kuvataan yksityiskohtaisesti tämän luvun aliluvuissa. Aliluvuissa kuvataan lisäksi testitapausten suoritusten eteneminen ja suoritusten tulokset.

6.1 Testaustilanteen kuvaus

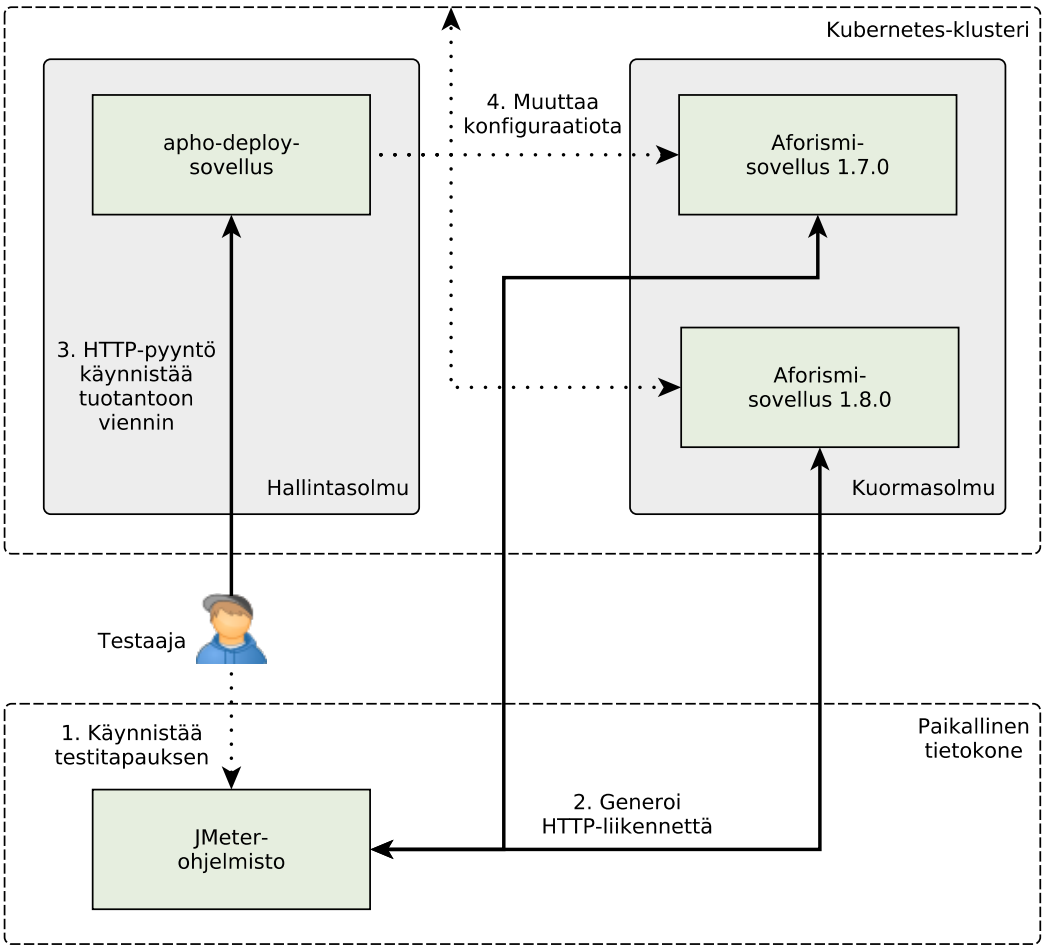
Testitapauksissa testattiin AphoDeploy-sovelluksen toimintaa suorittamalla vaiheittainen käyttöönotto Kubernetes-klusterissa AphoDeploy-sovelluksen avulla. Kuvassa 6.1 esitellään testaustilanteeseen liittyvät komponentit ja komponenttien väliset suhteet. Jokaista testitapausta suoritettiin kolme kertaa, jotta testitapausten suorituksista voitiin laskea keskiarvosuoritus. Testitapausten suoritusten tulokset aggregoitiin yhden sekunnin tarkkuuteen. Keskiarvosuoritusten tulokset on laskettu yhden sekunnin tarkkuuteen aggregoiduista suoritusten tuloksista.

Testaustilanne alkoi sen varmistamisella, että Kubernetes-klusterin tila vastasi testitapausten lähtötilannetta. Testaustilanteen aikana AphoDeploy-sovellusta suoritettiin Kubernetes-klusterin hallintasolmulla paikallisesti. Jos Kubernetes-klusterin tila poikkesi testitapausten lähtötilanteesta, pystyi klusterin tilan palauttamaan lähtötilanteeseen AphoDeploy-sovelluksen avulla.

Kun Kubernetes-klusterin tila vastasi testitapauksen lähtötilannetta, käynnistettiin testitapausta varten liikenteen generointi JMeter-ohjelmistossa[14]. JMeter-ohjel-

[14]https://jmeter.apache.org/ (Vierailtu 29.11.2019).

miston avulla generoitiin HTTP-liikennettä Aforismi-sovellukseen ja seurattiin Aforismi-sovelluksen lähettämiä vastauksia generoituun liikenteeseen. JMeter-ohjelmiston ”JSON Assertion”-komponenttia[15] käytettiin tarkistamaan, miltä Aforismi-sovelluksen versiolta vastaus generoituun liikenteeseen tuli.



Kuva 6.1: Testaustilanteen komponentit ja komponenttien väliset suhteet.

JMeter-ohjelmistolla generoidussa liikenteessä HTTP-pyyntöön asetettiin otsakkeiden ”Aphorism-User” ja ”Aphorism-Organization” arvot vastaamaan taulukossa 6.1 esitettyjä arvoja. Jokaisessa testitapauksessa oli käytössä kolme organisaatiota, joissa jokaisessa oli kolme käyttäjää, joiden liikennettä generoitiin JMeter-ohjelmiston avulla. Lisäksi JMeter-ohjelmiston avulla generoitiin liikennettä, joka vastasi AphoDeploy-sovelluksen lähettämää liikennettä.

JMeter-ohjelmiston liikenteen generoinnin käynnistämisen jälkeen, käynnistettiin testitapauksen mukainen vaiheittainen käyttöönotto tekemällä HTTP-pyyntö

[15]https://jmeter.apache.org/usermanual/component_reference.html (Vierailtu 11.12.2019).

Aphorism-Organization	**Aphorism-User**	**Aphorism-User**	**Aphorism-User**
apho-deploy-organization	apho-deploy		
A	A1	A2	A3
B	B1	B2	B3
C	C1	C2	C3

Taulukko 6.1: Testitapauksissa otsakkeen arvoina käytetyt organisaatiot ja organisaatioiden jäsenet.

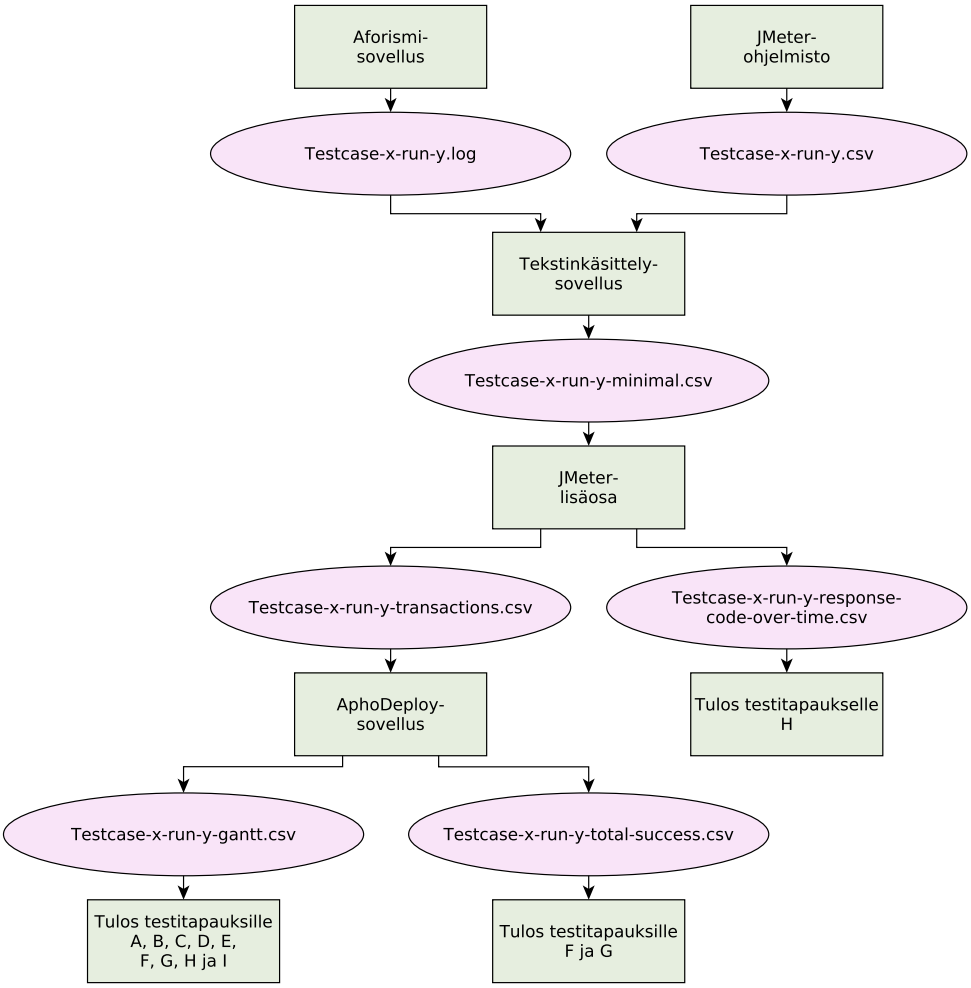
AphoDeploy-sovellukseen. Tämän jälkeen AphoDeploy-sovellus suoritti vaiheittaisen käyttöönoton muuttamalla Kubernetes-klusterin konfiguraatioita. Vaiheittaisen käyttöönoton etenemistä seurattiin AphoDeploy-sovelluksen logista. Vaiheittaisen käyttöönoton päätyttyä pysäytettiin liikenteen generointi JMeter-ohjelmistolla.

Kuvassa 6.2 esitellään testaustilanteessa käytetyt ohjelmistot ja testaustilanteen aikana muodostetut tiedostot. Testaustilanteen aikana muodostetuista tiedostoista muokattiin uusia tiedostoja testitapausten tulosten esittämistä varten. Tiedostojen nimissä on kuvassa käytetty kirjainta ”x” testitapauksen kirjaimen tilalla ja kirjainta ”y” testitapauksen suorituksen numeron tilalla.

Tiedostosta *Testcase-x-run-y-gantt.csv* on muodostettu tässä luvussa esitellyt testitapausten suorituskaaviot jokaiselle testitapaukselle. Tiedostosta *Testcase-x-run-y-total-success.csv* on muodostettu testitapauksille F ja G liikenteen kulun keskiarvokaaviot. Tiedostosta *Testcase-x-run-y-response-code-over-time.csv* on muodostettu testitapaukselle H HTTP-virhekoodien keskiarvomäärän näyttävä kaavio.

6.2 Tekniset vaatimukset

Seuraavissa alakohdissa käydään läpi testitapaukset, joilla testattiin vaiheittaisen käyttönoton menetelmälle asetettuja teknisiä vaatimuksia. Teknisiä vaatimuksia ovat ohjelmiston lähdekoodin muuttumattomuus, ohjelmistoversioiden rinnakkaisuus ja käyttökatkon pituus. Alakohdissa esitellään testitapausten A ja B sisältö, testitapausten suoritusten eteneminen ja tulokset.



Kuva 6.2: Testaustilanteessa käytetyt ohjelmistot ja testaustilanteen aikana muodostetut tiedostot.

6.2.1 Ohjelmiston lähdekoodin muuttumattomuus

Tieteellisessä kirjallisuudessa tuotannossa suoritettavan kokeilun toteutustekniikoiksi on ehdotettu muun muassa ominaisuusmuuttujia ja ajonaikaista liikenteenohjausta (Schermann et al., 2018). Ominaisuusmuuttujien avulla voidaan esimerkiksi ohjelmiston lähdekoodiin toteutetuilla ehdollisilla if-else-lauseilla sallia tietylle käyttäjäryhmälle pääsy vaihtoehtoiseen suorituspolkuun ohjelmistossa (Hodgson, 2017). Tämä tarkoittaa sitä, että ominaisuusmuuttujien käyttö vaatii muutoksia ohjelmiston lähdekoodiin.

Ajonaikaisessa liikenteenohjauksessa verkkoliikennettä ohjataan ohjelmiston suorituksen aikana tiettyyn ohjelmiston versioon (Schermann et al., 2018). Ajonaikainen liikenteenohjaus voidaan toteuttaa käyttämällä välityspalvelimia liikenteen ohjaamiseen esimerkiksi HTTP-pyyntöjen otsakkeiden arvojen perusteella.

Istion avulla liikennettä voidaan ohjata sekä otsakkeiden arvojen että prosenttiosuuden perusteella (Istio Authors, 2019d, 2019g). Liikenteenohjaus näitä tekniikoita käyttäen ei vaadi muutoksia ohjelmiston lähdekoodiin. Tämän vuoksi vaiheittaisen käyttöönoton toteutusmenetelmänä käytetään tässä tutkielmassa ajonaikaista liikenteenohjausta ominaisuusmuuttujien sijaan.

Esimerkissä 6.1 nähdään VirtualService-konfiguraatio, jonka avulla AphoDeploy-sovelluksen ja organisaation A jäsenten liikenne ohjataan Aforismi-sovelluksen versioon 1.8.0. Kaikki muu liikenne ohjataan Aforismi-sovelluksen versioon 1.7.0. Liikenteenohjaus tehdään käyttäen otsakkeiden ”Aphorism-User” ja ”Aphorism-Organization” tarkkoja arvoja. Liikenne tunnistetaan tulevan esimerkiksi organisaation A jäseneltä, jos otsakkeen ”Aphorism-Organization” tarkka arvo on ”A”. Kun käytetään VirtualService-konfiguraatiota liikenteen ohjaamiseen Aforismi-sovelluksen eri versioihin, Aforismi-sovelluksen lähdekoodiin ei ole tarvetta tehdä muutoksia. Tämä tarkoittaa, että vaatimus ohjelmiston lähdekoodin muuttumattomuudesta täyttyy.

6.2.2 Ohjelmistoversioiden rinnakkaisuus

Ohjelmistoversioiden rinnakkaisuusvaatimuksena on, että ohjelmistosta on voitava olla tuotannossa käytössä uusi ohjelmistoversio vanhan ohjelmistoversion rinnalla. Testitapaus A kuvaa yksinkertaisen tapauksen, jossa tuotannossa otetaan käyttöön uusi ohjelmistoversio vanhan ohjelmistoversion rinnalle.

Testitapaus A

Lähtötilanne:

- Klusterissa on käytössä Aforismi-sovelluksen versio 1.7.0, johon on ohjattu kaikki liikenne.
- Aforismi-sovelluksen versio 1.8.0 on käännettynä binäärirepositoriossa.

Vaiheittainen käyttöönotto:

1. Viedään klusteriin käyttöön versio 1.8.0 onnistuneesti.
2. Ohjataan AphoDeploy-sovelluksen liikenne versioon 1.8.0.
3. Seurataan liikenteen kulkua yhden minuutin ajan. AphoDeploy-sovelluksen liikenne kulkee onnistuneesti uuteen versioon 1.8.0.

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: aphorism-backend
  namespace: aphorism
spec:
  hosts:
    - "*"
  gateways:
    - aphorism-gateway
  http:
    #- name: "apho-deploy-routes"
    - match:
        - headers:
            aphorism-user:
              exact: apho-deploy
            aphorism-organization:
              exact: apho-deploy-organization
          uri:
            prefix: /aphorism/api/
      rewrite:
        uri: "/"
      route:
        - destination:
            host: aphorism-backend.aphorism.svc.cluster.local
            port:
              number: 80
            subset: 1-8-0
    #- name: "organization-a-routes"
    - match:
        - headers:
            aphorism-organization:
              exact: A
          uri:
            prefix: /aphorism/api/
      rewrite:
        uri: "/"
      route:
        - destination:
            host: aphorism-backend.aphorism.svc.cluster.local
            port:
              number: 80
            subset: 1-8-0
    #- name: "fallback-backend-routes"
    - match:
        - uri:
            prefix: /aphorism/api/
      rewrite:
        uri: "/"
      route:
        - destination:
            host: aphorism-backend.aphorism.svc.cluster.local
            port:
              number: 80
            subset: 1-7-0
```

Esimerkki 6.1: VirtualService-konfiguraatio liikenteen ohjaamista varten.

Onnistumiskriteerit:

- Aforismi-sovelluksen versiot 1.7.0 ja 1.8.0 ovat yhtä aikaa käytössä klusterissa.
- Versio 1.7.0 vastaanottaa organisaatioiden A, B ja C jäsenten liikenteen onnistuneesti.
- Versio 1.8.0 vastaanottaa AphoDeploy-sovelluksen liikenteen onnistuneesti.

Kuvassa 6.3 on kaavio testitapauksen A keskiarvoisesta suorituksesta. Kaavion vaaka-akselilla on näkyvissä testitapauksen suoritusaika, jonka kokonaiskesto on merkitty pystyakselille riville ”Kokonaiskesto”. Vaaka-akselin palkkien päälle on merkitty nimikkeeksi palkin pituus pyöristettynä kokonaisiksi minuuteiksi ja sekunneiksi. Vaaka-akselille piirrettyjen palkkien pituus vastaa täsmälleen palkin pituutta desimaalilukuna. Koska palkkien nimikkeinä käytetään pyöristettyjä minuutti- ja sekuntiarvoja, saattaa palkkien pituus joissain tapauksissa erota nimikkeiden arvoista.

Kuvassa 6.3 rivillä ”apho-deploy 1.8.0” kuvataan Aforismi-sovelluksen version 1.8.0 saavutettavuus AphoDeploy-sovelluksen näkökulmasta. Kuvasta voidaan havaita, että versiot 1.7.0 ja 1.8.0 ovat onnistuneesti yhtä aikaa käytössä klusterissa. Ajanhetken 01:06 jälkeen AphoDeploy-sovelluksen liikenne kulkee Aforismi-sovelluksen versioon 1.8.0 onnistuneesti. Lisäksi organisaatioiden A, B ja C jäsenten liikenne kulkee onnistuneesti versioon 1.7.0.

6.2.3 Käyttökatkon pituus

Käyttökatkon pituusvaatimuksena on, että ohjelmiston uuden version vaiheittaisen käyttöönoton aiheuttaman käyttökatkon tulee olla mahdollisimman lyhyt. Testitapauksessa B testataan tilannetta, jossa Aforismi-sovelluksen versio 1.8.0 otetaan käyttöön kaikille organisaatioille samanaikaisesti. Versio 1.8.0 otetaan käyttöön kaikille organisaatioille muuttamalla VirtualService-konfiguraatiota vaiheittaisen käyttöönoton aikana.

Testitapaus B

Lähtötilanne:

- Klusterissa on käytössä Aforismi-sovelluksen versio 1.7.0, johon on ohjattu kaikki liikenne.
- Aforismi-sovelluksen versio 1.8.0 on käännettynä binäärirepositoriossa.

Kuva 6.3: Testitapaus A, keskiarvosuoritus.

Vaiheittainen käyttöönotto:

1. Viedään klusteriin käyttöön versio 1.8.0 onnistuneesti.
2. Ohjataan kaikki liikenne versioon 1.8.0 ja poistetaan versioon 1.7.0 ohjattu liikenne.
3. Poistetaan versio 1.7.0 klusterista.
4. Seurataan liikenteen kulkua yhden minuutin ajan.

Onnistumiskriteerit:

- Aforismi-sovelluksen versio 1.8.0 on ajossa klusterissa.
- Liikenteenohjauksen vaihto versioon 1.8.0 ei aiheuta katkosta käyttäjille.
- Aforismi-sovelluksen versioon 1.7.0 ei saa lopuksi ohjautua liikennettä onnistuneesti.

Kuvasta 6.4 nähdään keskiarvosuoritus testitapaukselle B. Kuvasta nähdään, että ”VirtualService 1”-konfiguraatio viedään klusteriin keskimäärin ajanhetkellä 01:02. Tämän jälkeen kaikkien organisaatioiden jäsenten liikenne alkaa hetken päästä kulkea uuteen versioon 1.8.0 ja lakkaa kulkemasta versioon 1.7.0.

Tarkasteltaessa kuvasta esimerkiksi riviä ”A1 1.7.0” huomataan, että organisaation jäsenen A1 kohdalla onnistuneen ja epäonnistuneen vastauksen palkit leikkaavat. Tämä tarkoittaa, että keskimääräisesti jäsenen A1 pyynnöt Aforismi-sovelluksen versioon 1.7.0 onnistuvat ja epäonnistuvat yhtäaikaisesti hetken aikaa. Samanlainen tilanne havaitaan myös rivillä ”A1 1.8.0”, jossa ensin epäonnistuneet pyynnöt versioon 1.8.0 vaihtuvat onnistuneiksi pyynnöiksi.

Kuvaa tarkasteltaessa huomataan, että jäsenelle A1 ei tule katkoa siirryttäessä versiosta 1.7.0 käyttämään versiota 1.8.0. Rivien ”A1 1.7.0” ja ”A1 1.8.0” onnistuneiden vastausten palkit leikkaavat, mikä tarkoittaa, ettei versioiden vaihdon välillä keskimääräisesti ole katkoa.

Kaaviosta nähdään myös, että esimerkiksi jäsenellä A2 esiintyy keskimäärin katko siirryttäessä versiosta 1.7.0 versioon 1.8.0. Tämä voidaan havaita siitä, että riveillä ”A2 1.7.0” ja ”A2 1.8.0” onnistuneiden vastausten palkit eivät leikkaa. Jäsenen A2 katkon pituus jää kuitenkin kaavion perusteella vaatimuksissa mainitun kymmenen sekunnin rajan alle.

Kuva 6.4: Testitapaus B, keskiarvosuoritus.

6.3 Vaiheittaisen käyttöönoton vaatimukset

Seuraavissa alakohdissa esitellään vaiheittaisen käyttöönoton vaatimusten varmistamista varten toteutetut testitapaukset. Vaiheittaisen käyttöönoton vaatimuksia ovat ohjelmiston käyttäjäryhmän rajoittaminen, vaiheittaisen käyttöönoton kesto ja käyttäjämäärä käyttöönoton aikana. Alakohdissa esitellään testitapausten C, D, E, F ja G sisältö, testitapausten suoritusten eteneminen ja tulokset.

6.3.1 Ohjelmiston käyttäjäryhmän rajoittaminen

Ohjelmiston käyttäjäryhmän rajoittamista testaavassa testitapauksessa tiettyjen organisaatioiden jäsenten liikennettä ohjataan vaiheittain uuteen ohjelmistoversioon. Tällöin voidaan antaa uuteen ohjelmistoversioon pääsy vain tietyn organisaation jäsenille. Liikenteen ohjaaminen tehdään testitapauksessa C kahdessa vaiheessa VirtualService-konfiguraatioiden avulla. Ensimmäisessä vaiheessa A-organisaation jäsenten liikenne ohjataan uuteen versioon ”VirtualService 1”-konfiguraatiolla ja toisessa vaiheessa B-organisaation jäsenten liikenne ohjataan uuteen versioon ”VirtualService 2”-konfiguraatiolla.

Testitapaus C

Lähtötilanne:

- Klusterissa on käytössä Aforismi-sovelluksen versio 1.7.0, johon on ohjattu kaikki liikenne.
- Aforismi-sovelluksen versio 1.8.0 on käännettynä binäärirepositoriossa.

Vaiheittainen käyttöönotto:

1. Viedään klusteriin käyttöön versio 1.8.0 onnistuneesti.
2. Ohjataan organisaation A liikenne versioon 1.8.0.
3. Seurataan liikenteen kulkua yhden minuutin ajan. A-organisaation jäsenten liikenne kulkee versioon 1.8.0. B- ja C-organisaatioiden jäsenten liikenne kulkee versioon 1.7.0.
4. Ohjataan organisaation B liikenne versioon 1.8.0.
5. Seurataan liikenteen kulkua yhden minuutin ajan. A- ja B-organisaatioiden jäsenten liikenne kulkee versioon 1.8.0 ja C-organisaation jäsenten liikenne kulkee versioon 1.7.0.

Onnistumiskriteerit:

- Aforismi-sovelluksen versio 1.8.0 on käytössä klusterissa.
- Organisaatioiden A ja B jäsenten liikenne kulkee lopuksi versioon 1.8.0.
- Organisaation C jäsenten liikenne kulkee versioon 1.7.0.

Kuvasta 6.5 nähdään keskiarvosuoritus testitapaukselle C. Kuvasta voidaan havaita, että kaikilla A-organisaation jäsenillä liikenne ohjautuu ”VirtualService 1”-konfiguraation käyttöönoton jälkeen Aforismi-sovelluksen versioon 1.8.0. Lisäksi B-organisaation jäsenten liikenne ohjautuu uuteen versioon vasta ”VirtualService 2”-konfiguraation käyttöönoton jälkeen. C-organisaation jäsenten liikenne ohjautuu koko testin ajan Aforismi-sovelluksen versioon 1.7.0. Tämä tarkoittaa, että testitapauksen C onnistumiskriteerit täyttyvät.

6.3.2 Vaiheittaisen käyttöönoton kesto

”Vaiheittaisen käyttöönoton kesto”-vaatimuksena on, että uuden version käyttöönotto tulee voida suorittaa loppuun asti tietyn aikavälin aikana. Testitapauksessa D vaiheittainen käyttöönotto tehdään kolmessa vaiheessa ohjaamalla eri organisaatioiden jäsenten liikenne uuteen Aforismi-sovelluksen versioon vaiheittain yhden minuutin välein. Lopuksi kaikki liikenne ohjataan uuteen Aforismi-sovelluksen versioon. Testitapauksessa E liikenne ohjataan testitapauksen D tapaan vaiheittain uuteen versioon, mutta testitapauksessa E käytetään kahden minuutin väliaikaa liikenteen ohjaukseen tehtävien muutosten välissä.

Testitapaus D

Lähtötilanne:

- Klusterissa on käytössä Aforismi-sovelluksen versio 1.7.0, johon on ohjattu kaikki liikenne.
- Aforismi-sovelluksen versio 1.8.0 on käännettynä binäärirepositoriossa.

Vaiheittainen käyttöönotto:

1. Viedään klusteriin käyttöön versio 1.8.0 onnistuneesti.
2. Ohjataan organisaation A liikenne versioon 1.8.0.
3. Seurataan liikenteen kulkua yhden minuutin ajan. A-organisaation jäsenten liikenne kulkee versioon 1.8.0. B- ja C-organisaatioiden jäsenten liikenne kulkee versioon 1.7.0.
4. Ohjataan organisaation B liikenne versioon 1.8.0.
5. Seurataan liikenteen kulkua yhden minuutin ajan. A- ja B-organisaatioiden jäsenten liikenne kulkee versioon 1.8.0 ja C-organisaation jäsenten liikenne kulkee versioon 1.7.0.
6. Ohjataan organisaation C liikenne versioon 1.8.0.
7. Poistetaan versioon 1.7.0 ohjattu liikenne ja poistetaan versio 1.7.0 käytöstä.
8. Seurataan liikenteen kulkua yhden minuutin ajan. A-, B- ja C-organisaatioiden jäsenten liikenne kulkee versioon 1.8.0.

Kuva 6.5: Testitapaus C, keskiarvosuoritus.

Onnistumiskriteerit:

- Aforismi-sovelluksen versio 1.8.0 on käytössä klusterissa.
- Organisaatioiden A, B ja C jäsenten liikenne kulkee lopuksi versioon 1.8.0.
- Liikenteenohjauksen vaihtojen välissä on yhden minuutin väli.
- Aforismi-sovelluksen versioon 1.7.0 ei saa lopuksi ohjautua onnistuneesti liikennettä.

Kuvassa 6.6 nähdään keskiarvosuoritus testitapaukselle D. Keskiarvosuorituksesta nähdään, että ”VirtualService 3”-konfiguraation käyttöönoton jälkeen kaikkien organisaatioiden jäsenten liikenne ohjautuu Aforismi-sovelluksen versioon 1.8.0. Lisäksi VirtualService-konfiguraatioiden käyttöönottojen välillä on ensimmäisen käyttöönoton jälkeen yhden minuutin väli, jonka aikana liikenne kulkee viimeisimmän VirtualService-konfiguraation sääntöjen mukaan.

Testitapaus E

Lähtötilanne:

- Klusterissa on käytössä Aforismi-sovelluksen versio 1.7.0, johon on ohjattu kaikki liikenne.
- Aforismi-sovelluksen versio 1.8.0 on käännettynä binäärirepositoriossa.

Vaiheittainen käyttöönotto:

1. Viedään klusteriin käyttöön versio 1.8.0 onnistuneesti.
2. Ohjataan organisaation A liikenne versioon 1.8.0.
3. Seurataan liikenteen kulkua kahden minuutin ajan. A-organisaation jäsenten liikenne kulkee versioon 1.8.0. B- ja C-organisaatioiden jäsenten liikenne kulkee versioon 1.7.0.
4. Ohjataan organisaation B liikenne versioon 1.8.0.
5. Seurataan liikenteen kulkua kahden minuutin ajan. A- ja B-organisaatioiden jäsenten liikenne kulkee versioon 1.8.0 ja C-organisaation jäsenten liikenne kulkee versioon 1.7.0.
6. Ohjataan organisaation C liikenne versioon 1.8.0
7. Poistetaan versioon 1.7.0 ohjattu liikenne ja poistetaan versio 1.7.0 käytöstä.
8. Seurataan liikenteen kulkua kahden minuutin ajan. A-, B- ja C-organisaatioiden jäsenten liikenne kulkee versioon 1.8.0.

Onnistumiskriteerit:

- Aforismi-sovelluksen versio 1.8.0 on käytössä klusterissa.
- Organisaatioiden A, B ja C jäsenten liikenne kulkee lopuksi versioon 1.8.0.
- Liikenteenohjauksen vaihtojen välissä on kahden minuutin väli.
- Aforismi-sovelluksen versioon 1.7.0 ei saa lopuksi ohjautua onnistuneesti liikennettä.

Kuva 6.6: Testitapaus D, keskiarvosuoritus.

Kuvassa 6.7 nähdään keskiarvosuoritus testitapaukselle E. Testitapaus E eroaa testitapauksesta D siten, että VirtualService-konfiguraatioiden käyttöönottojen välillä on yhden minuutin välin sijaan kahden minuutin väli. Tämä nähdään myös kuvasta 6.7, jossa esimerkiksi ”VirtualService 2”-konfiguraation ja ”VirtualService 3”-konfiguraation käyttöönoton välillä on kahden minuutin väli. Käyttöönottovälin muuttaminen mahdollistaa vaiheittaisen käyttöönoton keston vaatimuksen täyttämisen.

6.3.3 Käyttäjämäärä vaiheittaisen käyttöönoton aikana

”Käyttäjämäärä vaiheittaisen käyttöönoton aikana”-vaatimuksessa määritellään, että ohjelmiston käyttäjämäärä tulee voida määrittää prosenttiosuutena ohjelmiston käyttäjistä. Testitapauksissa F ja G testataan tätä vaatimusta. Testitapauksessa F uuden version käyttäjämäärä määritellään prosenttiosuutena kaikkien käyttäjien liikenteestä. Testitapauksessa G yhdistetään liikenteen ohjausta organisaation perusteella prosenttiosuutena määriteltävään liikenteenohjaukseen.

Testitapaus F

Lähtötilanne:

- Klusterissa on käytössä Aforismi-sovelluksen versio 1.7.0, johon on ohjattu kaikki liikenne.
- Aforismi-sovelluksen versio 1.8.0 on käännettynä binäärirepositoriossa.

Vaiheittainen käyttöönotto:

1. Viedään klusteriin käyttöön versio 1.8.0 onnistuneesti.
2. Ohjataan 35% kaikesta liikenteestä versioon 1.8.0.
3. Seurataan liikenteen kulkua yhden minuutin ajan. Osan jäsenistä liikenne kulkee versioon 1.8.0 ja osan versioon 1.7.0.
4. Ohjataan 65% kaikesta liikenteestä versioon 1.8.0.
5. Seurataan liikenteen kulkua yhden minuutin ajan. Osan jäsenistä liikenne kulkee versioon 1.8.0 ja osan versioon 1.7.0.
6. Ohjataan 100% kaikesta liikenteestä versioon 1.8.0.
7. Poistetaan versioon 1.7.0 ohjattu liikenne ja poistetaan versio 1.7.0 käytöstä.
8. Seurataan liikenteen kulkua yhden minuutin ajan. Kaikkien jäsenten liikenne kulkee versioon 1.8.0.

Kuva 6.7: Testitapaus E, keskiarvosuoritus.

Onnistumiskriteerit:

- Aforismi-sovelluksen versio 1.8.0 on käytössä klusterissa.
- Kaikkien jäsenten liikenne kulkee lopuksi versioon 1.8.0.
- Liikenteenohjauksen muutokset tapahtuvat tiettyjen prosenttiosuuksien välein.
- Aforismi-sovelluksen versioon 1.7.0 ei saa lopuksi ohjautua onnistuneesti liikennettä.

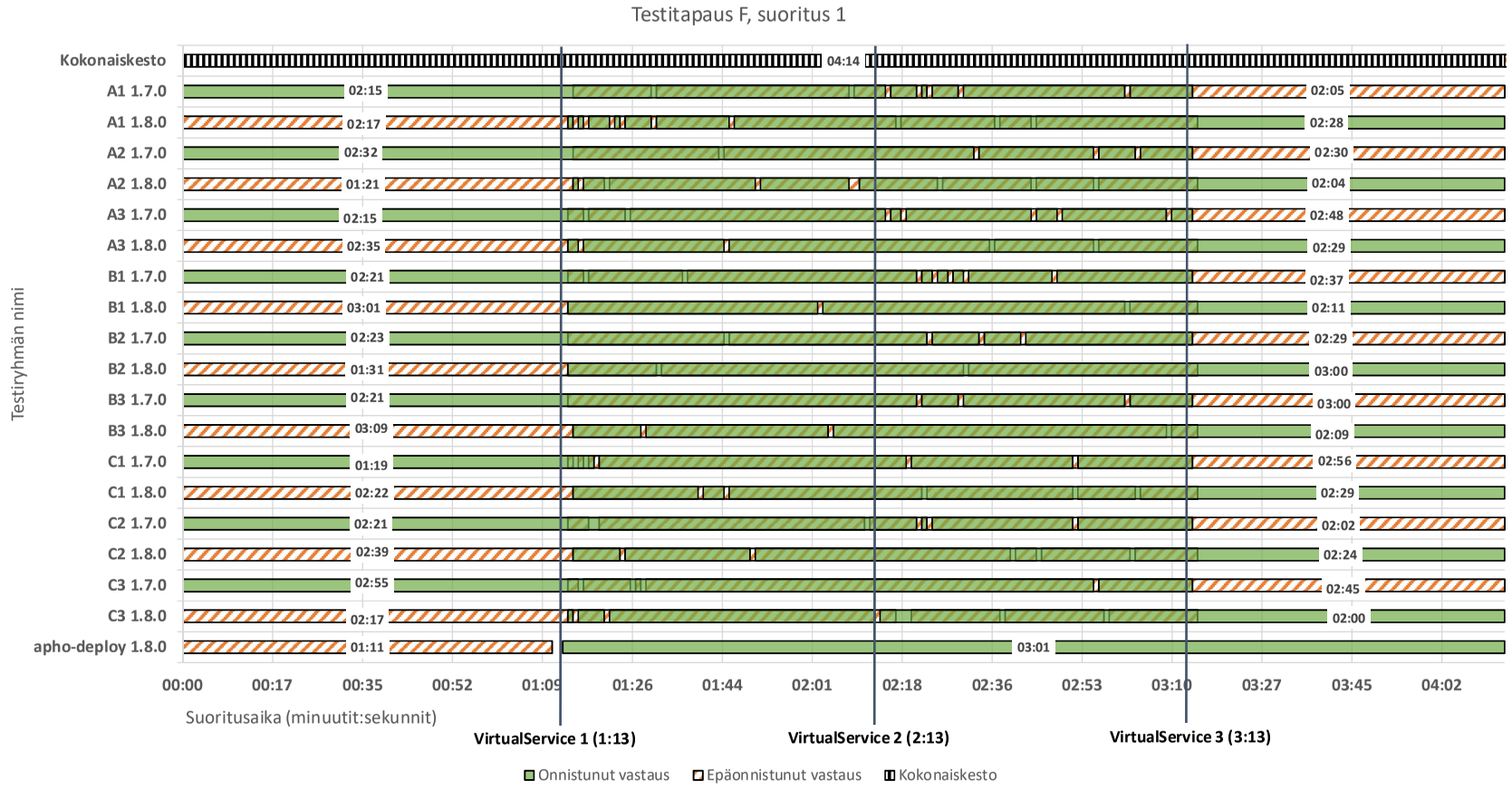
Kuvissa 6.8, 6.9 ja 6.10 nähdään testitapauksen F suoritusten tulokset. Suoritusten tuloksiin on merkitty jokaiselle riville testitapauksen alussa ja lopussa esiintyvä pisin yhtenäinen aika, jolloin tietyn jäsenen liikenne on onnistuneesti tai epäonnistuneesti kulkenut tiettyyn Aforismi-sovelluksen versioon. Kuvista voidaan havaita, että liikenteen eteneminen ”VirtualService 1”-konfiguraation ja ”VirtualService 3”-konfiguraation käyttöönottojen välillä vaihtelee testitapauksittain.

Missään suorituksessa Aforismi-sovelluksen versioon 1.8.0 ei ohjaudu liikennettä ennen ”VirtualService 1”-konfiguraation käyttöönottoa. Lisäksi jokaisessa suorituksessa ”VirtualService 3”-konfiguraation käyttöönoton jälkeen versioon 1.7.0 ohjautuva liikenne lakkaa pian konfiguraation käyttöönoton jälkeen, jolloin kaikki liikenne ohjautuu versioon 1.8.0.

”VirtualService 1”- ja ”VirtualService 3”-konfiguraatioiden käyttöönoton välillä liikenteen ohjautuminen versioihin 1.7.0 ja 1.8.0 vaihtelee. Esimerkiksi suorituksen 2 kuvasta 6.9 riveiltä ”A3 1.7.0” ja ”A3 1.8.0” nähdään ennen ”VirtualService 2”-konfiguraation käyttöönottoa, että jäsen A3 näkee yhtä aikaa sekä version 1.7.0 että version 1.8.0. Rivillä ”A3 1.8.0” kuvassa 6.9 onnistuneissa vastauksissa esiintyy myös katkoksia, joiden aikana versioon 1.8.0 ei ohjaudu liikennettä.

Testitapauksen F yksittäisten suoritusten kuvien avulla ei voida päätellä, kuinka monta prosenttia jäsenistä näkee version 1.7.0 tai version 1.8.0 tietyllä ajan hetkellä. Kuvien avulla voidaan päätellä, onko tietyn jäsenen liikennettä ohjautunut tiettyyn Aforismi-sovelluksen versioon tietyllä ajan hetkellä sekunnin tarkkuudella.

Kuvassa 6.11 nähdään keskiarvo testitapauksen F kolmen suorituksen aikaisesta liikenteen kulusta alkaen ajanhetkestä 01:00 ja päättyen ajanhetkeen 03:20. Kuvassa näkyy organisaatioiden A, B ja C jäsenten Aforismi-sovelluksen versioon 1.7.0 ja versioon 1.8.0 ohjautuvan onnistuneen liikenteen prosenttiosuus tietyllä ajan hetkellä.



Kuva 6.8: Testitapaus F, suoritus 1.

Kuva 6.9: Testitapaus F, suoritus 2.

Kuva 6.10: Testitapaus F, suoritus 3.

Esimerkiksi ajanhetkellä 01:30 keskimäärin 66% liikenteestä kulkee onnistuneesti versioon 1.7.0 ja 34% liikenteestä kulkee versioon 1.8.0.

Kuvan 6.11 perusteella voidaan nähdä, että VirtualService-konfiguraatioiden avulla ohjattu liikenne kulkee onnistumiskriteerien mukaisesti. Kolmen testitapauksen aikana ”VirtualService 1”-konfiguraatio otetaan käyttöön keskimäärin ajanhetkellä 01:08, ”VirtualService 2”-konfiguraatio ajanhetkellä 02:08 ja ”VirtualService 3”-konfiguraatio ajanhetkellä 03:09. Kuvasta 6.11 havaitaan, että näiden ajanhetkien jälkeen Aforismi-sovelluksen versioon 1.8.0 ohjautuvan liikenteen määrä kasvaa ja vastaavasti versioon 1.7.0 ohjautuvan liikenteen määrä pienentyy. Ajanhetkien 01:08 ja 02:08 välillä versioon 1.7.0 kulkee liikennettä pääasiassa yli 60%. Ajanhetkien 02:08 ja 03:09 välillä versioon 1.7.0 kulkee pääasiassa yli 30% liikenteestä.

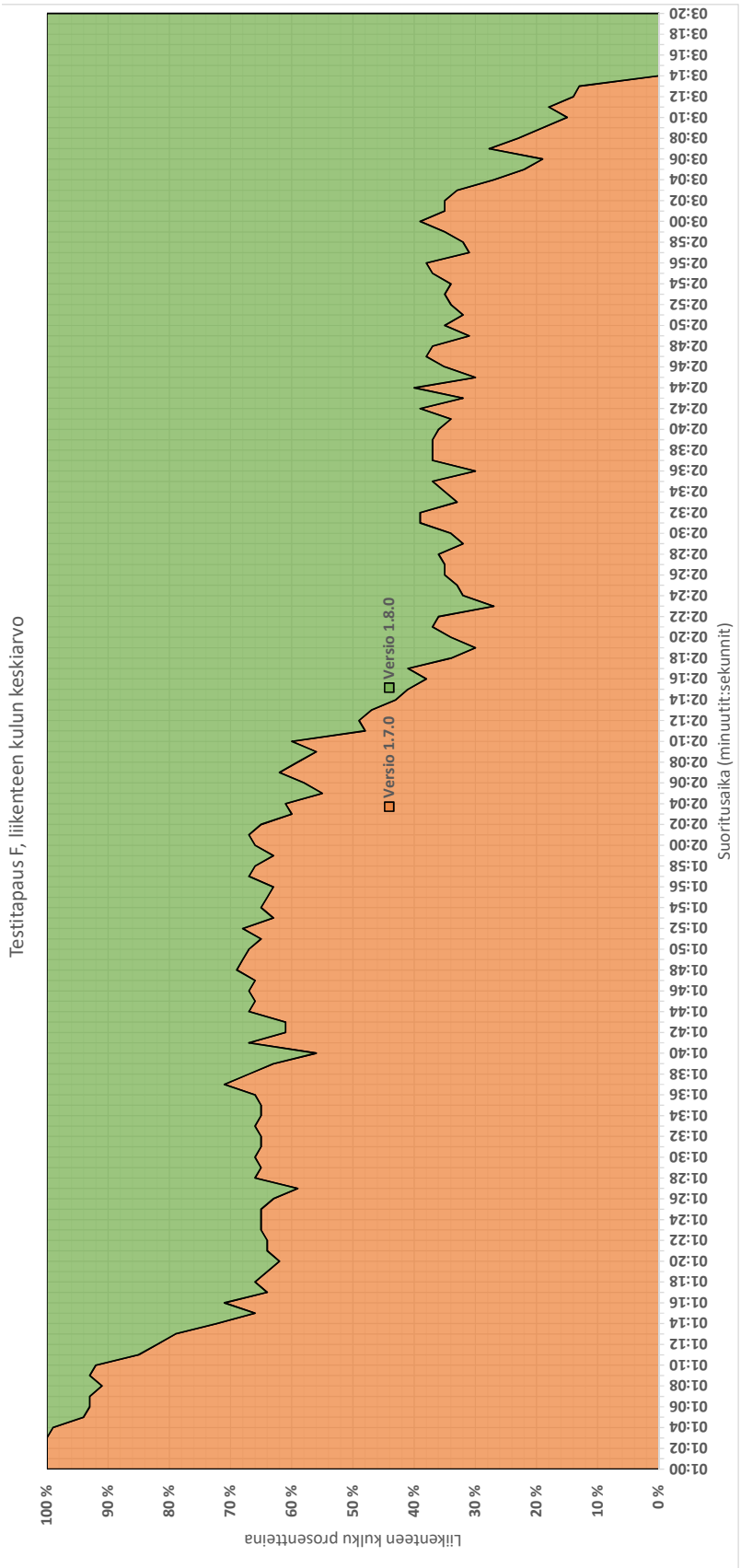
Testitapaus G

Lähtötilanne:

- Klusterissa on käytössä Aforismi-sovelluksen versio 1.7.0, johon on ohjattu kaikki liikenne.
- Aforismi-sovelluksen versio 1.8.0 on käännettynä binäärirepositoriossa.

Vaiheittainen käyttöönotto:

1. Viedään klusteriin käyttöön versio 1.8.0 onnistuneesti.
2. Ohjataan organisaation A liikenne versioon 1.8.0.
3. Seurataan liikenteen kulkua yhden minuutin ajan. A-organisaation jäsenten liikenne kulkee versioon 1.8.0. B- ja C-organisaatioiden jäsenten liikenne kulkee versioon 1.7.0.
4. Ohjataan 35% B- ja C-organisaatioiden liikenteestä versioon 1.8.0.
5. Seurataan liikenteen kulkua yhden minuutin ajan. A-organisaation jäsenten liikenne kulkee versioon 1.8.0 ja osan B- ja C-organisaatioiden jäsenten liikenteestä kulkee versioon 1.7.0 ja osan versioon 1.8.0.
6. Ohjataan 65% B- ja C-organisaatioiden liikenteestä versioon 1.8.0.
7. Seurataan liikenteen kulkua yhden minuutin ajan. A-organisaation jäsenten liikenne kulkee versioon 1.8.0 ja osan B- ja C-organisaatioiden jäsenten liikenteestä kulkee versioon 1.7.0 ja osan versioon 1.8.0.
8. Ohjataan 100% B- ja C-organisaatioiden liikenteestä versioon 1.8.0.
9. Poistetaan versioon 1.7.0 ohjattu liikenne ja poistetaan versio 1.7.0 käytöstä.
10. Seurataan liikenteen kulkua yhden minuutin ajan. A-, B- ja C-organisaatioiden jäsenten liikenne kulkee versioon 1.8.0.



Kuva 6.11: Testitapaus F, liikenteen kulun keskiarvo.

Onnistumiskriteerit:

- Aforismi-sovelluksen versio 1.8.0 on käytössä klusterissa.
- Organisaatioiden A, B ja C jäsenten liikenne kulkee lopuksi versioon 1.8.0.
- Liikenteenohjauksen muutokset voivat tapahtua sekä organisaation että prosenttiosuuden perusteella samanaikaisesti.
- Liikenteenohjauksen muutokset tapahtuvat tiettyjen prosenttiosuuksien välein.
- Aforismi-sovelluksen versioon 1.7.0 ei saa lopuksi ohjautua onnistuneesti liikennettä.

Kuvissa 6.12, 6.13 ja 6.14 nähdään testitapauksen G kolmen suorituksen tulokset. Testitapauksen suoritusten aikana klusteriin viedään käyttöön neljä VirtualService-konfiguraatiota. Kuvista nähdään, että ensimmäisen VirtualService-konfiguraation käyttöönoton jälkeen organisaation A jäsenten liikenne kulkee koko testin ajan Aforismi-sovelluksen versioon 1.8.0. Toisen ja neljännen VirtualService-konfiguraation käyttöönoton välillä B- ja C-organisaatioiden liikenne kulkee vaihtelevasti versioiden 1.7.0 ja 1.8.0 välillä. Tämä vastaa samaa tilannetta kuin testitapauksessa F. Lopuksi kaikkien organisaatioiden liikenne kulkee versioon 1.8.0.

Kuvista nähdään myös, että liikenteen ohjaaminen samanaikaisesti sekä organisaation että prosenttiosuuden perusteella on mahdollista. Testitapauksen G suorituskuvien 6.12, 6.13 ja 6.14 perusteella ”VirtualService 2”-konfiguraation ja ”VirtualService 4”-konfiguraation käyttöönottojen välillä liikenteen ohjautumisessa eri versioiden välillä ei esiinny yhtä paljon katkoja kuin testitapauksessa F. Esimerkiksi rivillä ”B2 1.7.0” jokaisessa suorituksessa Aforismi-sovelluksen versioon 1.7.0 ohjautuu liikennettä ”VirtualService 4”-konfiguraation käyttöönottoon asti.

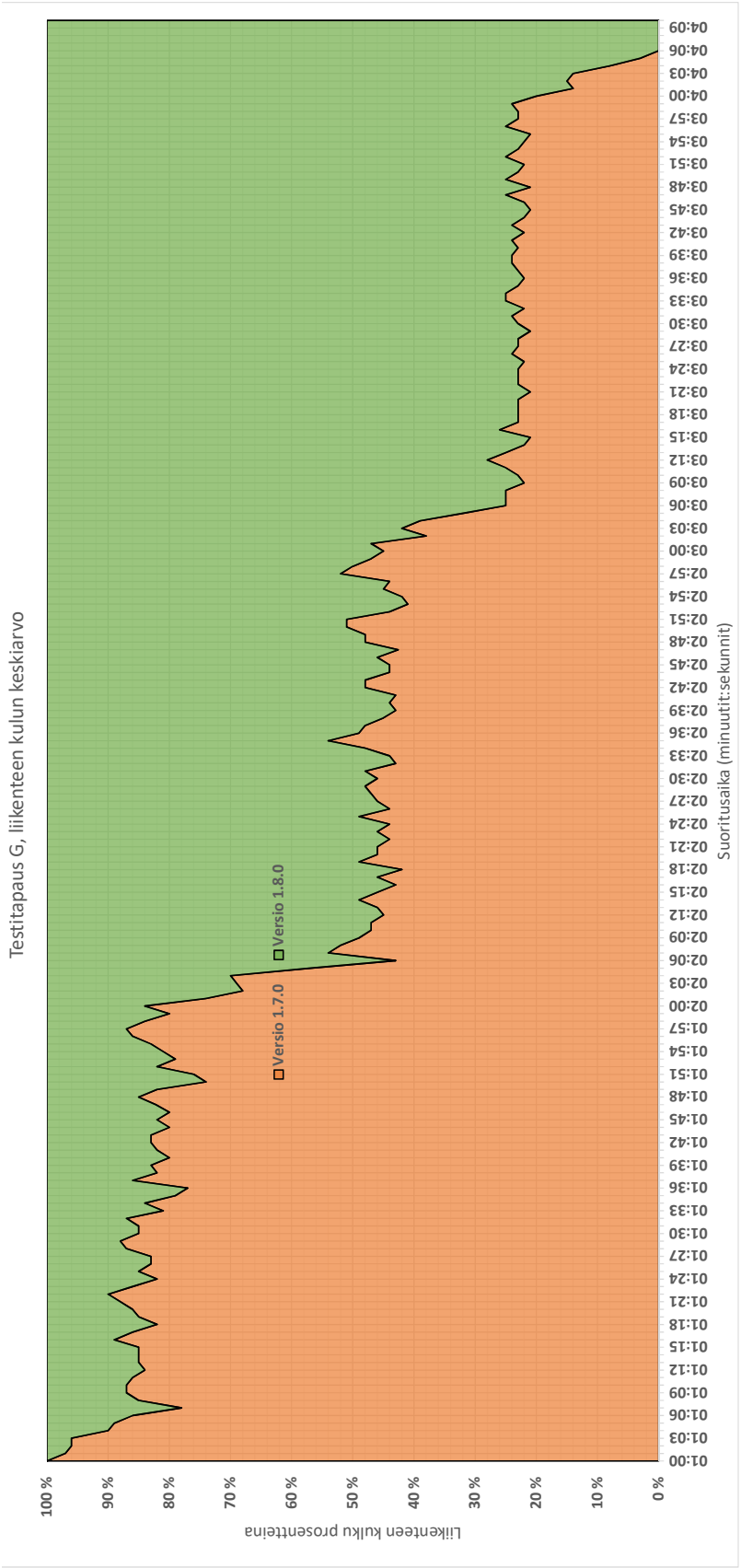
Kuvassa 6.15 nähdään keskiarvo testitapauksen G kolmen suorituksen aikaisesta liikenteen kulusta alkaen ajanhetkestä 01:00 ja päättyen ajanhetkeen 04:10. Kuvassa näkyy organisaatioiden A, B ja C jäsenten Aforismi-sovelluksen versioon 1.7.0 ja versioon 1.8.0 ohjautuvan onnistuneen liikenteen prosenttiosuus tietyllä ajan hetkellä.

Kuvan 6.15 perusteella voidaan nähdä, että VirtualService-konfiguraatioiden avulla ohjattu liikenne kulkee myös testitapauksessa G onnistumiskriteerien mukaisesti. ”VirtualService 1”-konfiguraation ja ”VirtualService 2”-konfiguraation välillä vain organisaation A liikenne ohjataan versioon 1.8.0. Tämä näkyy kuvassa keskimäärin

Kuva 6.12: Testitapaus G, suoritus 1.

Kuva 6.13: Testitapaus G, suoritus 2.

Kuva 6.14: Testitapaus G, suoritus 3.



Kuva 6.15: Testitapaus G, liikenteen kulun keskiarvo.

aikavälillä 01:04...02:04. Aikavälillä 02:04...03:04 on konfiguroitu lisäksi 35% B- ja C-organisaatioiden liikenteestä ohjautumaan versioon 1.8.0, mikä on myös nähtävissä kuvasta. Ajanhetken 03:04 jälkeen ensin 65% ja sitten 100% organisaatioiden B ja C liikenteestä ohjataan versioon 1.8.0.

6.4 Vaiheittaisen käyttöönoton päättyminen

Seuraavissa alakohdissa esitellään testitapaukset, joiden avulla testattiin vaiheittaisen käyttöönoton päättymiseen liittyviä vaatimuksia. Vaiheittaisen käyttöönoton päättymiseen liittyviä vaatimuksia ovat vanhan ohjelmistoversion poistaminen ja vaiheittaisen käyttöönoton keskeyttäminen. Alakohdissa esitellään testitapausten H ja I sisältö, testitapausten suoritusten eteneminen ja tulokset.

6.4.1 Vanhan ohjelmistoversion poistaminen

Vanhan ohjelmistoversion poistamiseen liittyvän vaatimuksen testaamisessa lähtötilanne, vaiheittainen käyttöönotto ja onnistumiskriteerit ovat samat kuin testitapauksessa B. Vanhan ohjelmistoversion poistamiseen liittyvän vaatimuksen toteutumista voidaan siis arvioida testitapauksen B perusteella.

Testitapauksesta B voidaan havaita, että vanhan ohjelmistoversion poistamisen vaatimus täyttyy. Testitapauksen B keskiarvosuorituksen esittelevästä kuvasta 6.4 havaitaan, että liikenteen ohjauksen muutoksen jälkeen kaikki liikenne kulkee Aforismi-sovelluksen versioon 1.8.0. Testitapauksen B suoritukseen kuuluu, että AphoDeploy-sovellus poistaa version 1.7.0 käytöstä automaattisesti.

6.4.2 Vaiheittaisen käyttöönoton keskeyttäminen

Ohjelmiston vaiheittainen käyttöönotto tulee voida keskeyttää, jos käyttöönoton aikana ohjelmiston toiminnassa havaitaan ongelmia. Testitapauksilla H ja I testataan tämän vaatimuksen täyttymistä. Testitapauksista ainoastaan H toteutuu nykyisessä AphoDeploy-sovelluksessa vaatimuksen mukaisesti, sillä testitapauksessa I esiintyvää ongelmaa ei voida nykyisen AphoDeploy-sovelluksen avulla havaita.

Testitapaus H

Lähtötilanne:

- Klusterissa on käytössä Aforismi-sovelluksen versio 1.7.0, johon on ohjattu kaikki liikenne.
- Aforismi-sovelluksen versio 1.9.0 on käännettynä binäärirepositoriossa.
- Aforismi-sovelluksen versiossa 1.9.0 on ongelma, joka näkyy vaiheittaisen käyttöönoton aikana.

Vaiheittainen käyttöönotto:

1. Viedään klusteriin ajoon versio 1.9.0.
2. Ohjataan AphoDeploy-sovelluksen liikenne versioon 1.9.0.
3. Versio 1.9.0 ei tule organisaatioiden jäsenille käyttöön ollenkaan, sillä vaiheittaisen käyttöönoton aikana havaitaan ongelma.
4. Palautetaan versio 1.7.0 käyttöön kaikille.
5. Poistetaan versio 1.9.0 ajosta klusterista.
6. Seurataan liikenteen kulkua yhden minuutin ajan. Kaikki liikenne kulkee versioon 1.7.0.

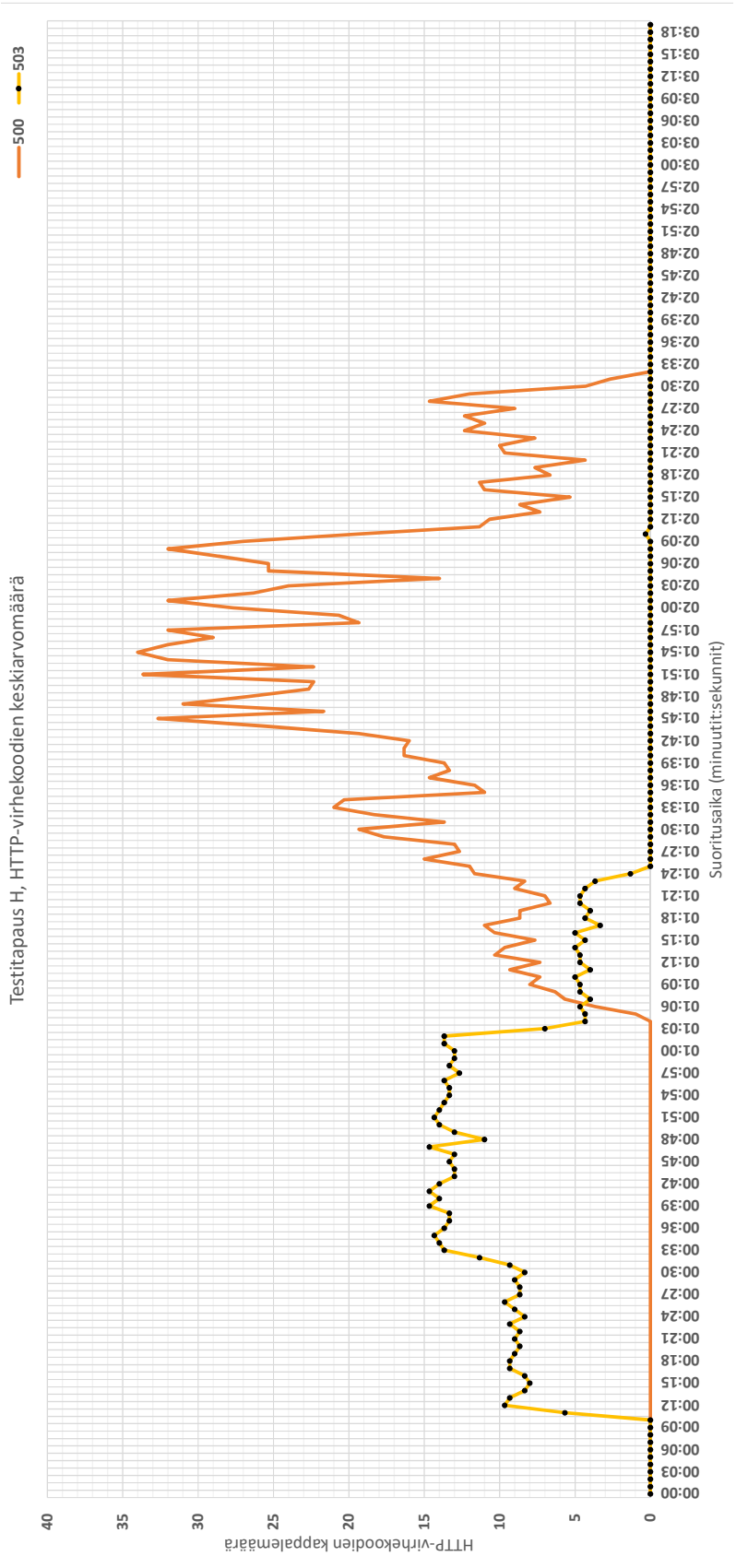
Onnistumiskriteerit:

- Aforismi-sovelluksen versioon 1.9.0 ei saa lopuksi ohjautua liikennettä ja versio 1.9.0 ei saa olla ajossa klusterissa.
- Organisaatioiden jäsenten liikennettä saa ohjautua vain Aforismi-sovelluksen versioon 1.7.0.
- Vaiheittaisen käyttöönoton aikana virheilmoituksia ei tule organisaatioiden jäsenille näkyviin.

Kuvasta 6.16 nähdään keskiarvosuoritus testitapaukselle H. Riviltä ”apho-deploy 1.9.0” huomataan, että AphoDeploy-sovelluksen liikenne Aforismi-sovelluksen versioon 1.9.0 epäonnistuu koko testitapauksen keskiarvosuorituksen ajan.

Kuvasta 6.17 nähdään, että ennen kuin versio 1.9.0 on valmiina vastaanottamaan liikennettä, palauttaa se HTTP-virhekoodin ”503 Service Unavailable”. HTTP-virhekoodi 503 tarkoittaa, että palvelu ei ole käytettävissä. Kun Aforismi-sovelluksen versio 1.9.0 on valmiina vastaanottamaan liikennettä, palauttaa se kuitenkin virhekoodin ”500 Internal Server Error”. HTTP-virhekoodi 500 tarkoittaa virhettä ohjelmiston toiminnassa. Kuvasta 6.17 huomataan, että virhekoodin 500 vastaanottaminen loppuu ennen testitapausten keskiarvosuorituksen loppua. Tämä tarkoittaa, että AphoDeploy-sovelluksen liikenne on siirretty takaisin Aforismi-sovelluksen versioon 1.7.0.

Kuva 6.16: Testitapaus H, keskiarvosuoritus.



Kuva 6.17: Testitapaus H, HTTP-virhekoodien keskiarvomäärä.

Testitapaus I

Lähtötilanne:

- Klusterissa on käytössä Aforismi-sovelluksen versio 1.7.0, johon on ohjattu kaikki liikenne.
- Aforismi-sovelluksen versio 1.9.1 on käännettynä binäärirepositoriossa.
- Aforismi-sovelluksen versiossa 1.9.1 on ongelma, joka näkyy organisaation A jäsenille.

Vaiheittainen käyttöönotto:

1. Viedään klusteriin käyttöön versio 1.9.1 onnistuneesti.
2. Ohjataan organisaation A liikenne versioon 1.9.1.
3. Seurataan liikenteen kulkua yhden minuutin ajan.
4. Havaitaan, että versiossa 1.9.1 on ongelma organisaation A jäsenille.
5. Palautetaan versio 1.7.0 käyttöön kaikille.
6. Poistetaan versio 1.9.1 käytöstä klusterista.

Onnistumiskriteerit:

- Aforismi-sovelluksen versioon 1.9.1 ei saa lopuksi ohjautua liikennettä ja versio 1.9.1 ei saa olla ajossa klusterissa.
- Lopuksi liikennettä saa ohjautua vain Aforismi-sovelluksen versioon 1.7.0.
- Vaiheittaisen käyttöönoton peruuttamisen jälkeen virheilmoituksia ei tule A-organisaation jäsenille näkyviin.

Kuvassa 6.18 nähdään testitapauksen I keskiarvoinen suoritus. Kuvasta nähdään riviltä ”apho-deploy 1.9.1”, että Aforismi-sovelluksen versio 1.9.1 otetaan onnistuneesti käyttöön. Lisäksi organisaation A jäsenten riveiltä nähdään, että ”VirtualService 1”-konfiguraation käyttöönoton jälkeen jäsenten A1, A2 ja A3 liikenne siirtyy pois versiosta 1.7.0. Organisaation A jäsenet saavat kuitenkin vain epäonnistuneita vastauksia Aforismi-sovelluksen versiolta 1.9.1 koko testitapauksen suorituksen ajan.

Testitapauksen I suoritukset epäonnistuvat vaiheittaisen käyttöönoton vaiheessa 4, jossa tulisi havaita, että versiossa 1.9.1 on ongelma organisaation A jäsenille. AphoDeploy-sovelluksen nykyisellä versiolla ei voida havaita tällaista tilannetta, sillä AphoDeploy-sovellus seuraa vain AphoDeploy-sovelluksesta lähetettyjen HTTP-pyyntöjen tilaa ja virhekoodeja.

Kuva 6.18: Testitapaus I, keskiarvosuoritus.

7 Pohdinta

Luvuissa 5 ja 6 esiteltiin menetelmä web-ohjelmiston vaiheittaiselle käyttöönotolle ja testitapaukset, joiden avulla menetelmää testattiin. Tässä luvussa esitellään menetelmää testanneiden testitapausten tulosten tulkinta ja mahdollisia tulevaisuuden tutkimuskohteita. Lisäksi luvussa käydään läpi tutkielman rajoitteet ja haasteet.

7.1 Tulosten tulkinta

Jatkuvaa julkaisua tutkivassa tieteellisessä kirjallisuudessa tehdään paljon yhteistyötä teollisuuden ja akatemian välillä (Auer & Felderer, 2018). Myös tässä tutkielmassa on tehty yhteistyötä kohdeyrityksen kanssa toteuttamalla menetelmä web-ohjelmiston vaiheittaiselle käyttöönotolle. Design science -tutkimusmenetelmän avulla on voitu yhdistää tieteellisessä kirjallisuudessa mainittuja käytäntöjä kohdeyrityksessä toteutettuun käytännön toteutukseen.

Tutkielman tutkimuskysymys oli, miten web-ohjelmiston vaiheittainen käyttöönotto voidaan toteuttaa käytännössä. Luvussa 5 esiteltiin web-ohjelmiston vaiheittaiselle käyttöönotolle vaatimukset, jotka vaiheittaisen käyttöönoton menetelmän tulisi toteuttaa. Vaiheittaisen käyttöönoton suorittamista varten toteutettiin vaiheittaisen käyttöönoton sovellus AphoDeploy. AphoDeploy-sovelluksen avulla Kubernetes-klusteriin, johon on asennettu palveluverkkoratkaisu Istio, voidaan viedä vaiheittain käyttöön uusi versio jo käytössä olevaan ohjelmistoon.

AphoDeploy-sovelluksen toimintaa testattiin luvussa 6 esitellyillä testitapauksilla. Taulukkoon 7.1 on koottu testitapaukset ja testitapausten tulokset. Testitapausten avulla voitiin osoittaa vaatimukset 2, 3, 4, 5, 6, ja 7 täytetyiksi. Vaatimus 8 täyttyi vain osittain, sillä AphoDeploy-sovelluksen nykyisellä toteutuksella ei havaita uudessa versiossa esiintyviä ongelmia, jos ne eivät ilmene AphoDeploy-sovelluksen tekemissä HTTP-pyynnöissä. Ohjelmiston lähdekoodin muuttumattomuuteen liittyvä vaatimus numero 1 voidaan myös todeta täyttyneeksi. AphoDeploy-sovelluksen käyttö vaiheittaisessa käyttöönotossa ei vaatinut muutoksia Aforismi-sovelluksen lähdekoodiin, kun käytettiin Istion VirtualService-konfiguraatioita ajonaikaiseen liikenteenohjaukseen.

Seitsemän vaiheittaiselle käyttöönotolle asetetusta vaatimuksesta täyttyi kokonaan ja yksi vaatimus täyttyi osittain käytettäessä AphoDeploy-sovellusta ohjelmiston

Testitapaus	Testaa vaatimusta	Vaatimuksen numero	Testin tulos
A	Ohjelmistoversioiden rinnakkaisuus	2	onnistui
B	Käyttökatkon pituus	3	onnistui
B	Vanhan ohjelmistoversion poistaminen	7	onnistui
C	Ohjelmiston käyttäjäryhmän rajoittaminen	4	onnistui
D	Vaiheittaisen käyttöönoton kesto	5	onnistui
E	Vaiheittaisen käyttöönoton kesto	5	onnistui
F	Käyttäjämäärä vaiheittaisen käyttöönoton aikana	6	onnistui
G	Käyttäjämäärä vaiheittaisen käyttöönoton aikana	6	onnistui
H	Vaiheittaisen käyttöönoton keskeyttäminen	8	onnistui
I	Vaiheittaisen käyttöönoton keskeyttäminen	8	epäonnistui

Taulukko 7.1: Vaiheittaisen käyttöönoton menetelmää testaavien testitapausten tulokset.

vaiheittaisessa käyttöönotossa. Näin ollen voidaan todeta, että tutkielmassa toteutettu vaiheittaisen käyttöönoton menetelmä on yksi tapa toteuttaa web-ohjelmiston vaiheittainen käyttöönotto. Tutkielmassa löydettiin siis vastaus tutkielman tutkimuskysymykseen.

Tutkimuskysymykseen vastaamisen lisäksi AphoDeploy-sovellus tarjoaa kohdeyritykselle mahdollisuuden automatisoidun vaiheittaisen käyttöönoton jatkokehittämiseen. Tutkimuksen aikana kehitetty AphoDeploy-sovellus luo siis pohjaa työkaluinfrastruktuurille, joka voisi tulevaisuudessa tukea sekä regressiopohjaista että liiketoimintapohjaista jatkuvaa kokeilua. AphoDeploy-sovelluksen tarjoama HTTP-rajapinta voisi mahdollistaa AphoDeploy-sovelluksen yhdistämisen myös osaksi tuotantoon viennin putkea. Jotta AphoDeploy-sovelluksen yhdistäminen osaksi putkea olisi mahdollista, tulisi AphoDeploy-sovellusta jatkokehittää.

AphoDeploy-sovellus on toteutettu suorittamaan vaiheittainen käyttöönotto Aforismi-sovellukselle. Vaihtamalla AphoDeploy-sovelluksessa käytetyt Aforismi-sovelluksen VirtualService- ja DestinationRule-konfiguraatiot toisen ohjelmiston konfiguraatioihin, mahdollistaa AphoDeploy-sovellus automatisoidun vaiheittaisen käyttöönoton myös muille ohjelmistoille. VirtualService- ja DestinationRule-konfiguraatioita

muuttamalla on siis mahdollista konfiguroida myös muille ohjelmistoille vaiheittainen käyttöönotto esimerkiksi HTTP-pyyntöjen otsakkeiden arvojen perusteella tai haluttujen prosenttiosuuksien suuruisissa vaiheissa. AphoDeploy-sovelluksen VirtualService-konfiguraatioiden käyttöönoton ajastuksia muuttamalla on myös mahdollista konfiguroida vaiheittainen käyttöönotto tapahtumaan halutun pituisten vaiheiden aikana.

Jotta AphoDeploy-sovellus olisi nykyistä helpommin hyödynnettävissä yleisesti myös muiden ohjelmistojen vaiheittaiseen käyttöönottoon, tulisi AphoDeploy-sovellusta jatkokehittää. Jatkokehityksen aikana AphoDeploy-sovellukseen voitaisiin lisätä mahdollisuus määritellä vaiheittaisessa käyttöönotossa käytettävät VirtualService- ja DestinationRule-konfiguraatiot ja haluttu vaiheittaisen käyttöönoton kulku esimerkiksi JSON-muodossa. JSON-muotoinen konfiguraatio voitaisiin lähettää AphoDeploy-sovellukselle vaiheittaisen käyttöönoton käynnistämisen yhteydessä HTTP-pyynnöllä. Lisäksi vaiheittaisen käyttöönoton vaiheiden kestoa ja järjestystä tulisi voida säätää vapaasti konfiguraation avulla.

Jatkokehityksen aikana AphoDeploy-sovellukseen tulisi myös yhdistää klusteriympäristön valvonta. Valvonnan yhdistämisen avulla AphoDeploy-sovelluksella käynnistetyn vaiheittaisen käyttöönoton aikana voitaisiin nykyistä monipuolisemmin havaita klusteriympäristön mahdolliset ongelmatilanteet. Myös tieteellisessä kirjallisuudessa on raportoitu valvonnan hyödyntämisestä päätöksenteossa, kun ajonaikaisen liikenteenohjauksen konfiguraatioita muutetaan (Schermann et al., 2016). Jos AphoDeploy-sovelluksella olisi valvonnan kautta mahdollisuus havaita klusteriympäristön ongelmatilanteet, voitaisiin testitapauksessa I havaittu puute korjata. Tällöin AphoDeploy-sovelluksella käynnistetyssä vaiheittaisessa käyttöönotossa voitaisiin toipua automaattisesti myös testitapauksen I kaltaisista ongelmatilanteista.

7.2 Tutkielman rajoitteet ja haasteet

Tutkielmassa toteutettuun AphoDeploy-sovellukseen ja AphoDeploy-sovelluksen toimintaa testaaviin testitapauksiin liittyy rajoitteita. Rajoitteet vaikuttavat siihen, miten laajasti tutkielmassa saadut tulokset ovat hyödynnettävissä ja ovatko tutkielmasta saadut tulokset yleistettävissä. Yksi tutkielman rajoitteista on, että tässä tutkielmassa on käsitelty vaiheittaista käyttöönottoa vain web-ohjelmiston osalta. Tämä tarkoittaa, että tutkielman tulokset eivät ole yleistettävissä sulautettuihin

järjestelmiin tai työpöytäohjelmistoihin. Sulautettujen järjestelmien tai työpöytäohjelmistojen kohdalla tässä tutkielmassa esitelty vaiheittaisen käyttöönoton menetelmä ei välttämättä ole käytettävissä.

Toinen tutkielman rajoitteista on, että Aforismi-sovellus on tilaton ohjelmisto. Tämä tarkoittaa, että tutkielmassa esitellyn vaiheittaisen käyttöönoton menetelmän soveltuvuudesta tilallisten ohjelmistojen vaiheittaiseen käyttöönottoon ei voida tehdä päätelmiä tämän tutkielman perusteella. Tutkielman perusteella on havaittu, että vaiheittaisen käyttöönoton aikana ohjelmiston käyttäjän näkemä ohjelmistoversio saattaa muuttua. Ohjelmistoversion muuttumisella voisi olla vaikutuksia myös käyttäjän näkemään ohjelmiston tilaan.

Kolmantena rajoitteena tutkielmassa on, että tutkielmassa tutkittiin vaiheittaista käyttöönottoa vain Kubernetes-klusterissa, johon asennettiin tutkielman aikana palveluverkkoratkaisu Istio. Tämän vuoksi tutkielman aikana toteutetun vaiheittaisen käyttöönoton menetelmän sopivuudesta muihin ympäristöihin ei ole varmuutta.

Yksi tulevaisuuden tutkimuskohteista voisi olla selvittää, sopiiko tutkielmassa käytetty vaiheittaisen käyttöönoton menetelmä myös muihin ympäristöihin. Tutkielmassa vaiheittaisen käyttöönoton menetelmän pohjana käytettiin tieteellisessä kirjallisuudessa esiteltyä menetelmää. Menetelmässä käytetään välityspalvelimia ajonaikaisen liikenteenohjauksen toteuttamiseen ja erillistä ohjelmistoa liikenteenohjauksen konfiguraation muuttamiseen. Tämä viittaa siihen, että tutkielmassa toteutettu vaiheittaisen käyttöönoton menetelmä voisi soveltua myös muissa ympäristöissä käytettäväksi.

Neljäntenä rajoitteena tutkielmassa on, että tutkielmassa ei tutkittu tutkielman aikana toteutetun vaiheittaisen käyttöönoton menetelmän vaikutusta ohjelmistojen suorituskykyyn. Tällaisten vaikutusten olemassa olon selvittäminen voisi olla yksi tulevaisuuden tutkimuskohteista. Yksi selvitettävistä asioista voisi olla, miten liikenteen ohjaaminen HTTP-pyyntöjen otsakkeiden arvojen perusteella tai liikenteen prosenttiosuuden perusteella vaikuttaa pyyntöjen vasteaikoihin.

Viidentenä rajoitteena tutkielmassa ovat ohjelmistot, joiden ohjelmistopäivitysten aikana muutetaan ohjelmiston käyttämän tietokannan rakennetta. Aforismi-sovelluksen toteutukseen ei kuulunut tietokantaa. AphoDeploy-sovelluksen soveltuvuutta tietokantaa käyttävien ohjelmistojen vaiheittaiseen käyttöönottoon ei ole siis tässä tutkielmassa tutkittu. Jos ohjelmistosta on tuotannossa käytössä samanaikaisesti

useita versioita, jotka hyödyntävät yhtä aikaa samaa tietokantaa, tulisi eri versioiden pystyä toimimaan tietokannan kanssa, vaikka tietokannan rakenne muuttuisi uusien ohjelmistopäivitysten yhteydessä. Tietokannan tilan ja sisällön hallitseminen saattaa siis aiheuttaa ongelmia, joita tässä tutkielmassa ei ole huomioitu vaiheittaisen käyttöönoton menetelmässä.

Tietokannan tilan palauttaminen ongelmatilanteissa aiempaan versioon on myös yksi tutkielman rajoitteista. Nykyisessä AphoDeploy-sovelluksen versiossa ongelmatilanteissa käytetyssä aiemman ohjelmistoversion palautustavassa ei ole huomioitu tietokannan tilaa tai tietokannan tilan palauttamista.

Yhtenä tutkimukseen liittyvänä haasteena tutkielmassa on, että tutkielman kirjoittaja on toteuttanut yksin sekä käytännön toteutuksen vaiheittaiselle käyttöönotolle että vaiheittaista käyttöönottoa testaavat testitapaukset. Tämä voi aiheuttaa sen, että AphoDeploy-sovellusta testaavat testitapaukset eivät ole riittävän kattavia. Testitapauksista saattaa esimerkiksi puuttua negatiivisia testitapauksia, jotka olisivat osoittaneet AphoDeploy-sovelluksen sopimattomaksi menetelmäksi vaiheittaiselle käyttöönotolle. Tämä haaste on yritetty huomioida tutkielmassa ottamalla mukaan yksi epäonnistuva testitapaus.

Toinen tutkimukseen liittyvistä haasteista on, että AphoDeploy-sovellusta testaavien testitapausten suoritukset vaihtelevat jokaisella suorituskerralla. Suoritusten vaihtelu saattaa aiheuttaa testitapausten tulosten vääristymistä tai muuttumista. Kubernetes-klusterissa voi kestää vaihteleva aika, kunnes klusterissa käyttöönotetut ohjelmistot ja konfiguraatiot ovat käytettävissä, mikä voi aiheuttaa testitapausten suoritusten tulosten vaihtelua. Lisäksi verkkoliikenne saattaa aiheuttaa testitapausten suoritusten vaihtelua.

Jotta suoritusten vaihtelun vaikutusta testitapausten tuloksiin saatiin tutkielmassa pienennettyä, suoritettiin jokaista testitapausta kolme kertaa. Testitapausten suorituksista laskettiin tämän jälkeen keskiarvoiset suoritukset, joita käytettiin testitapausten tulosten tulkinnassa. Testitapauksista F ja G laskettiin keskiarvoiset tulokset kuitenkin vain liikenteen kulusta testitapausten aikana. Tällöin testitapausten F ja G suorituksissa esiintyneet yksilölliset vaihtelut nähdään yksittäisten suoritusten kaavioista.

8 Yhteenveto

Tässä tutkielmassa kehitettiin menetelmä web-ohjelmiston uuden version vaiheittaiselle käyttöönotolle ympäristöön, jossa vaatimuksena on palvelun käytön katkottomuus. Tutkielman tutkimuskysymyksenä oli, miten web-ohjelmiston vaiheittainen käyttöönotto voidaan toteuttaa käytännössä. Tutkielmassa tehtiin yhteistyötä yritykseltäyritykselle-toimialalla toimivan kohdeyrityksen kanssa ja käytettiin tutkimusmenetelmänä design science -tutkimusmenetelmää.

Uuden version vaiheittaiselle käyttöönotolle asetettiin kahdeksan vaatimusta, jotka jaettiin teknisiin vaatimuksiin, vaiheittaisen käyttöönoton vaatimuksiin ja vaiheittaisen käyttöönoton päättymisen vaatimuksiin. Vaatimuksia voitiin käyttää varmistettaessa, onko tutkielman aikana kehitetty vaiheittaisen käyttöönoton menetelmä sopiva web-ohjelmiston vaiheittaiseen käyttöönottoon kohdeyrityksessä.

Tutkielman aikana kehitettiin AphoDeploy-sovellus, jonka avulla vaiheittainen käyttöönotto voitiin automatisoida Kubernetes-klusterissa. Kubernetes-klusteriin asennettiin palveluverkkoratkaisu Istio, jonka tarjoamien Envoy-välityspalvelinten avulla klusterissa tehtiin ajonaikaista liikenteenohjausta. AphoDeploy-sovelluksen avulla muutettiin automatisoidusti Istion ja Kubernetes-klusterin konfiguraatioita tietyillä ajan hetkillä.

Vaiheittaisen käyttöönoton menetelmää arvioitiin sekä analysoimalla AphoDeploy-sovelluksen toimintaa että testitapauksilla, joiden avulla selvitettiin, toteuttaako vaiheittaisen käyttöönoton menetelmä sille asetetut kahdeksan vaatimusta. Testitapausten ja AphoDeploy-sovelluksen analysoinnin perusteella voitiin todeta, että vaiheittaiselle käyttöönotolle asetetuista vaatimuksista seitsemän vaatimusta täyttyi kokonaan ja yksi vaatimus täyttyi osittain.

Tutkielmassa löydettiin tapa toteuttaa web-ohjelmiston vaiheittainen käyttöönotto käytännössä AphoDeploy-sovelluksen avulla. Näin ollen tutkielmassa löydettiin vastaus tutkielman tutkimuskysymykseen. Tutkielman aikana kehitetyllä AphoDeploy-sovelluksella luotiin myös pohjaa työkaluinfrastruktuurille, joka voisi tulevaisuudessa tukea kohdeyrityksessä esimerkiksi jatkuvaa kokeilua. AphoDeploy-sovellusta voitaisiin myös jatkokehittää, jotta AphoDeploy-sovellus olisi nykyistä monipuolisemmin hyödynnettävissä eri ohjelmistojen vaiheittaisessa käyttöönotossa.

Vaikka AphoDeploy-sovellus toteutti kokonaan suurimman osan vaiheittaiselle käyttöönotolle asetetuista vaatimuksista, liittyy AphoDeploy-sovelluksen käyttöön automatisoidussa vaiheittaisessa käyttöönotossa myös rajoitteita. AphoDeploy-sovelluksen rajoitteet ja tutkielmaan liittyvät haasteet vaikuttavat siihen, miten laajasti tutkielmassa saadut tulokset ovat hyödynnettävissä ja yleistettävissä.

Tulevaisuudessa AphoDeploy-sovellusta jatkokehittämällä voitaisiin korjata AphoDeploy-sovelluksessa havaittuja rajoitteita. Tulevaisuuden tutkimuskohteena voisi olla selvittää, sopiiko tutkielmassa kehitetty vaiheittaisen käyttöönoton menetelmä Kubernetes-klusterin lisäksi käytettäväksi myös muissa ympäristöissä. Lisäksi tulevaisuudessa voitaisiin selvittää, vaikuttaako AphoDeploy-sovelluksen käyttö vaiheittaisessa käyttöönotossa ohjelmistojen suorituskykyyn ja soveltuuko AphoDeploy-sovellus tilallisten ohjelmistojen vaiheittaiseen käyttöönottoon.

Lähteet

Auer, F., & Felderer, M. (2018, Aug). Current State of Research on Continuous Experimentation: A Systematic Mapping Study. Teoksessa *2018 44th euromicro conference on software engineering and advanced applications (seaa)* (s. 335-344). doi: 10.1109/SEAA.2018.00062

Budinsky, F. (2018, June). *Canary Deployments using Istio.* Tarkistettu saatavilla `https://istio.io/blog/2017/0.1-canary/` (Accessed 26.12.2019.)

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016, tammikuuta). Borg, Omega, and Kubernetes. *Queue*, *14*(1), 10:70–10:93. Tarkistettu saatavilla `http://doi.acm.org/10.1145/2898442.2898444` doi: 10.1145/2898442.2898444

Docker Inc. (2019). *What is a Container?* Tarkistettu saatavilla `https://www.docker.com/resources/what-container` (Accessed 28.12.2019.)

Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176 - 189. Tarkistettu saatavilla `http://www.sciencedirect.com/science/article/pii/S0164121215001430` doi: https://doi.org/10.1016/j.jss.2015.06.063

Fowler, M. (2006, May). *Continuous Integration.* Tarkistettu 22.10.2019, saatavilla `https://martinfowler.com/articles/continuousIntegration.html`

Hevner, A., R, A., March, S., T, S., Park, Park, J., ... Sudha (2004, 03). Design Science in Information Systems Research. *Management Information Systems Quarterly*, *28*, 75-.

Hilton, M., Tunnell, T., Huang, K., Marinov, D., & Dig, D. (2016). Usage, Costs, and Benefits of Continuous Integration in Open-source Projects. Teoksessa *Proceedings of the 31st ieee/acm international conference on automated software engineering* (s. 426–437). New York, NY, USA: ACM. Tarkistettu saatavilla `http://doi.acm.org/10.1145/2970276.2970358` doi: 10.1145/2970276.2970358

Hodgson, P. (2017, Oct). *Feature Toggles (aka Feature Flags).* Tarkistettu 16.04.2019, saatavilla `https://martinfowler.com/articles/feature-toggles.html`

Humble, J. (2010, Aug). *Continuous Delivery vs Continuous Deployment.* Tarkistettu 19.08.2019, saatavilla https://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation* (1st p.). Addison-Wesley Professional.

Istio Authors. (2019a). *Architecture.* Tarkistettu saatavilla https://istio.io/docs/ops/deployment/architecture/ (Accessed 11.12.2019.)

Istio Authors. (2019b). *Destination Rule.* Tarkistettu saatavilla https://istio.io/docs/reference/config/networking/destination-rule/ (Accessed 28.12.2019.)

Istio Authors. (2019c). *Gateway.* Tarkistettu saatavilla https://istio.io/docs/reference/config/networking/gateway/ (Accessed 28.12.2019.)

Istio Authors. (2019d). *Request Routing.* Tarkistettu saatavilla https://istio.io/docs/tasks/traffic-management/request-routing/ (Accessed 11.12.2019.)

Istio Authors. (2019e). *Traffic Management.* Tarkistettu saatavilla https://istio.io/docs/concepts/traffic-management/ (Accessed 26.12.2019.)

Istio Authors. (2019f). *Traffic Management Best Practices.* Tarkistettu saatavilla https://istio.io/docs/ops/best-practices/traffic-management/ (Accessed 11.12.2019.)

Istio Authors. (2019g). *Traffic Shifting.* Tarkistettu saatavilla https://istio.io/docs/tasks/traffic-management/traffic-shifting/ (Accessed 13.12.2019.)

Istio Authors. (2019h). *Virtual Service.* Tarkistettu saatavilla https://istio.io/docs/reference/config/networking/virtual-service/ (Accessed 28.12.2019.)

Istio Authors. (2019i). *What is Istio?* Tarkistettu saatavilla https://istio.io/docs/concepts/what-is-istio/ (Accessed 2.12.2019.)

Leppänen, M., Mäkinen, S., Pagels, M., Eloranta, V., Itkonen, J., Mäntylä, M. V., & Männistö, T. (2015, Mar). The highways and country roads to continuous deployment. *IEEE Software*, *32*(2), 64-72. doi: 10.1109/MS.2015.50

Li, W., Lemieux, Y., Gao, J., Zhao, Z., & Han, Y. (2019, April). Service Mesh: Challenges, State of the Art, and Future Research Opportunities. Teoksessa *2019 ieee international conference on service-oriented system engineering (sose)* (s. 122-1225). doi: 10.1109/SOSE.2019.00026

Lindgren, E., & Münch, J. (2016). Raising the odds of success: the current state of experimentation in product development. *Information and Software Technology*, *77*, 80 - 91. Tarkistettu saatavilla `http://www.sciencedirect.com/science/article/pii/S0950584916300647` doi: https://doi.org/10.1016/j.infsof.2016.04.008

Morgan, W. (2017, April). *What's a service mesh? And why do I need one?* Tarkistettu 27.12.2019, saatavilla `https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/`

Neely, S., & Stolt, S. (2013, Aug). Continuous Delivery? Easy! Just Change Everything (Well, Maybe It Is Not That Easy). Teoksessa *2013 agile conference* (s. 121-128). doi: 10.1109/AGILE.2013.17

Olsson, H., Bosch, J., & Alahyari, H. (2013, 03). Towards R&D as Innovation Experiment Systems: A Framework for Moving Beyond Agile Software Development. *IASTED Multiconferences - Proceedings of the IASTED International Conference on Software Engineering, SE 2013*. doi: 10.2316/P.2013.796-008

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45-77. Tarkistettu saatavilla `https://doi.org/10.2753/MIS0742-1222240302` doi: 10.2753/MIS0742-1222240302

Rahman, M. T., Querel, L.-P., Rigby, P. C., & Adams, B. (2016). Feature Toggles: Practitioner Practices and a Case Study. Teoksessa *Proceedings of the 13th international conference on mining software repositories* (s. 201–211). New York, NY, USA:

ACM. Tarkistettu saatavilla http://doi.acm.org/10.1145/2901739.2901745 doi: 10.1145/2901739.2901745

Rissanen, O., & Münch, J. (2015, May). Continuous Experimentation in the B2B Domain: A Case Study. Teoksessa *2015 ieee/acm 2nd international workshop on rapid continuous software engineering* (s. 12-18). doi: 10.1109/RCoSE.2015.10

Rodríguez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., ... Oivo, M. (2017). Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software*, *123*, 263 - 291. Tarkistettu saatavilla http://www.sciencedirect.com/science/article/pii/S0164121215002812 doi: https://doi.org/10.1016/j.jss.2015.12.015

Schermann, G., Cito, J., Leitner, P., Zdun, U., & Gall, H. C. (2018). We're doing it live: A multi-method empirical study on continuous experimentation. *Information and Software Technology*, *99*, 41 - 57. Tarkistettu saatavilla http://www.sciencedirect.com/science/article/pii/S0950584917302136 doi: https://doi.org/10.1016/j.infsof.2018.02.010

Schermann, G., Schöni, D., Leitner, P., & Gall, H. C. (2016). Bifrost: Supporting Continuous Deployment with Automated Enactment of Multi-Phase Live Testing Strategies. Teoksessa *Proceedings of the 17th international middleware conference* (s. 12:1–12:14). New York, NY, USA: ACM. Tarkistettu saatavilla http://doi.acm.org/10.1145/2988336.2988348 doi: 10.1145/2988336.2988348

Shahin, M., Babar, M. A., & Zhu, L. (2016). The Intersection of Continuous Deployment and Architecting Process: Practitioners' Perspectives. Teoksessa *Proceedings of the 10th acm/ieee international symposium on empirical software engineering and measurement* (s. 44:1–44:10). New York, NY, USA: ACM. Tarkistettu saatavilla http://doi.acm.org/10.1145/2961111.2962587 doi: 10.1145/2961111.2962587

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access*, *5*, 3909-3943. doi: 10.1109/ACCESS.2017.2685629

The Kubernetes Authors. (2019a). *Controllers.* Tarkistettu saatavilla https://kubernetes.io/docs/concepts/architecture/controller/ (Accessed 28.12.2019.)

The Kubernetes Authors. (2019b). *Deployments.* Tarkistettu saatavilla https://kubernetes.io/docs/concepts/workloads/controllers/deployment/ (Accessed 28.12.2019.)

The Kubernetes Authors. (2019c). *Kubernetes Components.* Tarkistettu saatavilla https://kubernetes.io/docs/concepts/overview/components/ (Accessed 11.12.2019.)

The Kubernetes Authors. (2019d). *Managing Resources.* Tarkistettu saatavilla https://kubernetes.io/docs/concepts/cluster-administration/manage-deployment/ (Accessed 28.12.2019.)

The Kubernetes Authors. (2019e). *Namespaces.* Tarkistettu saatavilla https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ (Accessed 28.12.2019.)

The Kubernetes Authors. (2019f). *Pod Overview.* Tarkistettu saatavilla https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/ (Accessed 28.12.2019.)

The Kubernetes Authors. (2019g). *Service.* Tarkistettu saatavilla https://kubernetes.io/docs/concepts/services-networking/service/ (Accessed 28.12.2019.)

The Kubernetes Authors. (2019h). *Understanding Kubernetes Objects.* Tarkistettu saatavilla https://kubernetes.io/docs/concepts/overview/working-with-objects/kubernetes-objects/ (Accessed 28.12.2019.)

The Kubernetes Authors. (2019i). *What is Kubernetes.* Tarkistettu saatavilla https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ (Accessed 28.12.2019.)

Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015). Large-scale Cluster Management at Google with Borg. Teoksessa *Proceedings of the*

tenth european conference on computer systems (s. 18:1–18:17). New York, NY, USA: ACM. Tarkistettu saatavilla `http://doi.acm.org/10.1145/2741948.2741964` doi: 10.1145/2741948.2741964